トランジスタ技術

エレクトロニクスの基礎と実用技術を濃縮したフィールド・ワーク・マガジン

SPECIAL

No.72
2000 AUTUMN

PCを使いこなすためのファイル・フォーマットとデータ転送I/F

特集

パソコン周辺インターフェースのすべてIII

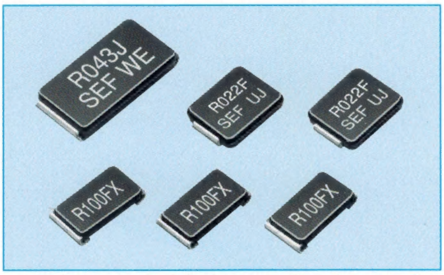
SEMITEC®

電流センサ
CURRENT SENSORS

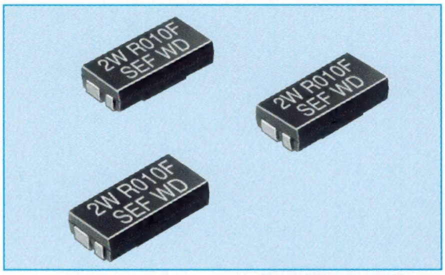
当社の電流センサは、磁気抵抗素子や電流トランスを使用する方式に比べて、非常に簡便、安価な低抵抗の抵抗器です。抵抗体に銅ニッケル合金を使用しているために温度係数が小さく安定し過度電力に強い特長をもっております。

■特長

- ●使用温度範囲が広い(−55℃〜155℃、−55℃〜180℃)
- ●抵抗温度係数が小さい(±100ppm/℃以下、±50ppm/℃以下もあり)
- ●熱膨張、収縮に強い
- ●サージ限界電力が大きい
- ●周波数特性が良い
- ●寸法精度が良く、耐衝撃性に優れている
- ●端子精度、半田付け性に優れている



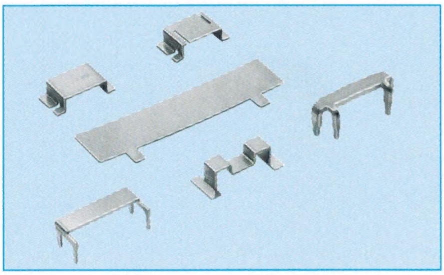
SRS、SRP、SRLシリーズ



SRFシリーズ(四端子品)

■仕様(SRS、SRP、SRL、SRFシリーズ)

- ●抵抗値許容差は、D(±0.5%)から制作可能です
- ●小型、軽量でお客様のニーズに応えた面実装の抵抗器です
- ●抵抗値範囲 3.0mΩ〜1.8Ω
- ●定格電力 1、2、3Wタイプがあります



SP、PWシリーズ

■仕様(SP、PWシリーズ)

- ●高負荷条件に耐えられる抵抗器です
- ●SP、PWシリーズは、カスタム品のためお客様のニーズに合わせて制作可能です
- ●抵抗値範囲 2.0mΩ〜10Ω
- ●電流値範囲 1A〜20A

■用途、応用製品例

バッテリーパック保護、残量表示回路、DC−DCコンバータ、インバータ回路、自動車用ヘッドライト、スイッチング電源、測定器、家電製品、電動工具、ファン、オートミラー、パワーウィンドウ、工作機械 等

ホームページでONLINE CATALOG(PDF)公開中……URL http://www.semitec.co.jp/
E-mailでのお問い合わせはこちら……………webmaster@mail.semitec.co.jp

SEMITEC® 石塚電子株式会社

本　社／〒130-8512 東京都墨田区錦糸1-7-7　●営業本部ダイヤルイン　☎03-3621-2703　<FAX> 03-3623-6100
西日本／〒532-0003 大阪市淀川区宮原2-1-17 (新大阪プロスペルビル)　☎06-6391-6491(代表)　<FAX> 06-6395-3649
名古屋／〒465-0095 名古屋市名東区高社1-263 (一社中央ビル)　☎052-777-5070(代表)　<FAX> 052-777-5061

IT時代のキー・テクノロジは…マイクロプロセッサとネットワーク

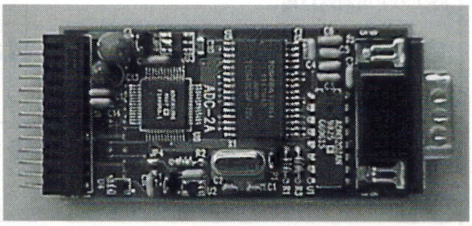
一味違った電子教材とハードウエアを提供します

8チャネル12ビットADコンバータ搭載
マイクロコンバータ基板

TRX-CV0 7,000円
パラレル入出力24ポート

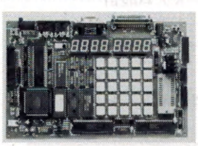
TRX-CV1 15,000円
外部４Ｍ SRAM付き基板

- 単三電池2本で動作可能…アナログ・デバセズ社ADuC812搭載
- わかりやすいCISC命令…8051をコアにした最新ワンチップ・マイコン
- 無償開発ツール…アセンブラ、デバッガ、ANSI-Cコンパイラ評価版をWEBで配布
- 書き込み器(ライタ)一切不要…8KバイトEEPROMとダウンローダ内蔵
- 256バイトSRAM、640バイトフラッシュEEPROM内蔵
- 外部4M SRAM搭載(TRX-CV1)
- 8チャネル12ビットA-Dコンバータ、2チャネル12ビットD-Aコンバータを搭載
- UARTシリアル・コントローラ…1チャネル(ISPダウンローダ兼用)
- 3チャネル16ビット・タイマ/カウンタ
- 32本のプログラマブル入出力
- 基板サイズ 75mm×37.5mm



作りながら学ぶZ80トレーニング基板
TRNジュニア

- 基本キット(A) 12,000円
- 基本+拡張キット(A+B) 21,000円
- 完成品(A+B) 26,000円

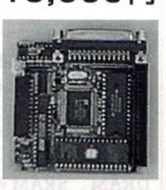


Z80系TMPZ84C013、ROM 32K、RAM 32K、25キー、8桁LED、セントロ、RS232C、AD、DA変換、MIDI、120曲オルゴール、BASICおよびZビジョン対応、資料集

RISCに迫る40MHｚ相当Z80…KL5C80A16
カードCPU

TRX-CP6 18,000円

吉田幸作氏が執筆のために設計したCPU基板
川崎製鉄製Z80互換プロセッサKL5C80A16搭載
オンボード書込み可能なEEPROM搭載でROM化は5秒、拡張I/OコネクタはTRNジュニアと互換。
大容量高速メモリ(1M EPROM,1M SRAM)搭載。
4チャネル16ビット・タイマ/カウンタ、2チャネル・シリアル、DMA、24ビットパラレルI/O



TRNジュニア・カードCPU基板互換(完成品)
拡張入出力基板TRXシリーズ

- 5V電源基板 4,800円
- パラレル入出力基板 7,800円
- シリアル入出力基板 7,800円
- フォトカプラ基板 7,800円
- タイマ/カウンタ基板 7,800円
- パワードライブ基板 7,800円
- SSR基板 7,800円
- PWM制御基板 8,200円
- ステッピングモータ基板 7,800円
- 8軸DCモータ駆動基板 9,800円
- FM&GM128音源基板 14,000円
- 16ch12ビットAD基板 18,000円
- 4ch12ビットDA基板 18,000円
- マザーボード、TMPZ84C015基板

もの作りの心を育てる…
電子工作キットTRAシリーズ

便利になりすぎた現在…もの作りの楽しさを体験しよう

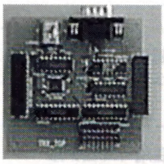
- リズムサウンド基板 2,000円
- 赤外リモコン ねずみロボット基板 7,000円
- 3色LED表示基板 98,000円
- OPL4 GM音源基板 13,000円
- ロボット制御基板 48,000円
- 音声効果音基板 2,500円
- 飲酒チェッカ・キット 14,800円
- ボイスレコーダ・キット 9,000円
- DOS/V用MIDIアダプタ 3,800円
- 積分型デジタル電圧計 3,000円

TCP/IPがハードウエア(LSI)になった
TCP/IP基板

TRX-TCP 9,000円

セイコーインスツルメンツ社が世界ではじめて製品化したTCP/IPプロトコルLSI、S-7600Aを搭載した拡張基板。Z80、H8、PICクラスのマイクロプロセッサでTCP/IP環境が実現可能。デジタル家電や制御機器のネットワーク化に最適。3.3V電源で動作します。

- 3.3V電源基板 4,800円
- 3.3V電源KL5C80A16基板 18,000円



64ビットIDコード内蔵1線式バスシステム
MicroLANコンポーネント

- COMポート・アダプタ 6,000円
- 4ch16ビットA-D基板 4,800円
- 3端子温度計 DS1820 1,100円
- 1ビットデジタル入出力 700円

パソコンでロジック設計からチップ書込み
MACH開発セット 30,000円

マイクロプロセッサと電子部品
(株)ESP企画
FAX. 0774-65-1321
E-mail oficejo@esp.co.jp URL http://www.esp.co.jp

◆価格はすべて税・送料込みです。
◆ご注文は現金書留もしくは銀行振込でお願いします。
◆入金後2週間以内に納品します。

A 銀行振込み
大垣共立銀行 羽島支店
普通 ８０２６９０
名義 ㈱イーエスピー企画

B 現金書留
〒501−6226
岐阜県羽島市正木町三ッ柳123
(株)ESP企画 江崎雅美

資料請求 No.5

エレクトロニクスの基礎と実用技術を濃縮した　フィールド・ワーク・マガジン

トランジスタ技術SPECIAL

季刊 ●B5判 ●定価 1〜33 =1,570円，34〜45 =1,631円，45〜49 =1,723円，50〜57 =1,835円，58 以降1,840円

1 個別半導体素子 活用法のすべて
基礎からマスタするダイオード，トランジスタ，FETの実用回路技術

6 Z80ソフト＆ハードのすべて
基礎からマクロ命令を使いこなすまでのノウハウを集大成

8 データ通信技術のすべて
シリアル・インターフェースの基礎からモデムの設計法まで

9 パソコン周辺機器インターフェース詳解
セントロニクス/RS-232C/GPIB/SCSIを理解するために

17 OPアンプによる回路設計入門
アナログ回路の誤動作とトラブルの原因を解く

22 ディジタル回路ノイズ対策技術のすべて
TTL/CMOS/ECLの活用法と誤動作/トラブルへの処方

25 最新マイコン・メモリ・システム設計法
DRAM，SRAMの動作からデュアルポートRAM，FIFOの活用まで

28 最新・電源回路設計技術のすべて
3端子レギュレータから共振型スイッチング電源まで

29 マイコン独習Z80完全マニュアル
手作りの原点から実用ソフトの作成まで

32 実用電子回路設計マニュアル
アナログ回路の設計例を中心に実用回路を詳述

36 基礎からの電子回路設計ノート
トランジスタ回路の設計からビデオ画像の編集まで

37 実用電子回路設計マニュアルII
豊富な回路設計例から最適設計を学ぼう

38 Z80システム設計完全マニュアル
周辺I/Oボードの設計とマイコン・システムの開発

39 A-Dコンバータの選び方・使い方のすべて
アナログ信号をディジタル処理するための基礎技術

40 電子回路部品の活用ノウハウ
機器の性能と信頼性を支える受動部品の使い方

41 実験で学ぶOPアンプのすべて
汎用OPアンプから高性能OPアンプまで

42 高速ディジタル回路の測定とトラブル解析
ハイスピード・ディジタル信号を高周波と捉える

44 フィルタの設計と使い方
アナログ回路のキーポイントを探る

45 PC98シリーズのハードとソフト
386＆486マシンを使いこなす！

47 高周波システム＆回路設計
通信新時代の回路技術とシステム設計

48 作れば解るCPU
ロジックICで実現するZ80とキャッスル・マシン

49 徹底解説 Z80マイコンのすべて
Z80CPUの概要から周辺LSIの活用法，ICEによるデバッグまで

50 フレッシャーズのための電子工学講座
電磁気学の基礎から電子回路の設計，製作までをやさしく解説

51 データ通信技術基礎講座
RS232Cの徹底理解からローカル通信の実用技術まで

52 ビデオ信号処理の徹底研究
映像信号処理の基礎から高画質化のためのディジタル信号処理の方法まで

53 パソコンによる計測・制御入門
研究室や実験室で必要なデータ収集のノウハウを基礎から解説

54 実践パワー・エレクトロニクス入門
パワーMOS FETとIGBTの使い方をやさしく解説

55 作ってわかる電子工作制作入門
やさしい電子工作からパソコンを使ったシステム開発まで

56 電子回路シミュレータ活用マニュアル
アナログ回路解析だけでなくディジタル回路解析も追加された

57 最新・スイッチング電源技術のすべて
効率とノイズを重点的に解説したソフト・スイッチングの指南書

58 基本・C-MOS標準ロジックIC活用マスタ
低電圧動作とドライブ能力の向上をはかった

59 新世代Z80CPUで学ぶマイコン入門
RISCライクなZ80互換プロセッサKC80を詳解する

60 実験で学ぼう回路技術のテクニック
オシロスコープの波形を見て，抵抗，コンデンサの使い方を覚えよう

61 モータ制御＆メカトロ技術入門
いろいろなモータとその駆動法を理解しよう

62 電子回路シミュレータの本格活用法
実証済み回路集で学ぶ設計のテクニック

63 パソコン周辺インターフェースのすべて
PCを使いこなすためのハードウェア規格リファレンス

64 実験で学ぶノイズ対策技術のすべて
回路をちゃんと動作させるために必要な知識を身につけよう

65 PCIバスの基礎と応用
Windows95パソコンの構成からCompactPCIシステムの構築まで

66 センサ応用回路の活用ノウハウ
基本的なセンサの使い方から応用回路設計まで

67 パソコン周辺機器インターフェースII
ATA(IDE)，SCSI，AGPを中心とした

68 WindowsPCによる計測・制御入門
パソコンを使ってデータ収集，解析を行う前に考えること

69 作ってわかる通信ネットワーク技術
携帯電話や自作LANを利用して情報収集をしよう

70 IEEE1394で広がる通信技術
パソコン周辺機器から情報家電のインターフェースまで

71 OPアンプから始めるアナログ技術
OPアンプ回路の設計からアナログPLDの活用まで

定価は税込

CQ出版社　〒170-8461　東京都豊島区巣鴨1-14-2　販売部　TEL(03)5395-2141　振替 00100-7-10665

特集

PCを使いこなすためのファイル・フォーマットとデータ転送I/F

パソコン周辺インターフェースのすべてIII

パソコン通信はプロバイダ経由のインターネットやメール送受信へと発展してきました．初めは通信内容が文字だけだったものが，画像や音声を文字情報とともに送りたいという要求が増え，パソコンの機種を問わないでこれらのデータを交換するための標準のファイル・フォーマットが必要になりました．今回は，画像フォーマットとして JPEG，TIFF，DIB，GIF，PNG，音声フォーマットとして WAVE，MP3 を取り上げ，それぞれの成り立ちからおおよその概要までを解説します．また，レガシー・インターフェースとしてパラレル・ポートとシリアル・ポート，PS/2 キーボード，PS/2 マウスの概要を解説します．

トランジスタ技術SPECIAL No.72

CONTENTS

特集 PCを使いこなすためのファイル・フォーマットとデータ転送I/F
パソコン周辺インターフェースのすべてIII

特集執筆：宮崎 仁

はじめに 本特集の読み方 ……… 4

ディジタル画像フォーマット（ラスタ・グラフィックスとデータ圧縮，ベクタ・グラフィックス，汎用ベクタ・グラフィックスの標準化）／ディジタル音声フォーマット

もっとも広く使われているディジタル画像の圧縮/符号化方式

第1-1章 JPEG ……… 8

ディジタル画像データを圧縮する必要性から生まれた規格／JPEG規格の成り立ちと規格化の過程／JPEG－連続階調の静止画データを圧縮／符号化する方式／画像データの形式/JPEG規格書の構成／圧縮/符号化方式にはどういうものがあるか／符号化のためのベースライン・プロセス/JPEGファイルの基本フォーマット／JFIFが規定する共通フォーマット

静止画データの画像フォーマット

第1-2章 TIFF ……… 37

周辺機器から取り込んだ画像データを利用するためのファイル・フォーマット／ベースラインTIFFと拡張TIFF（Rev. 6.0の仕様）／ベースラインTIFFの構成／TIFFのデータ圧縮 ― パック・ビット圧縮，HM圧縮，LZW圧縮，JPEG圧縮

Windowsで標準として使われている画像フォーマット

第1-3章 DIB ……… 57

BMPファイルはWindowsプログラミングの標準画像フォーマット／GUIのために作られたDIB画像フォーマットの成り立ち／標準構成はBITMAPINFOHEADER，RGBQUAD，BYTEの三つの部分／BITMAPFILEHEADER構造体の構成／BITMAPINFOHEADER構造体の構成／RGBQUAD構造体の構成／画像データの配列／画像の圧縮

コンピュータ間で画像ファイルをやりとりする画像交換フォーマット

第1-4章 GIF ……… 66

パソコン通信の時代からもっとも広く使われている画像ファイル形式／画像交換フォーマットGIFの誕生／GIFのライセンス問題と新しい画像ファイル・フォーマットPNG／GIFファイルの定義／GIFの画像であることを示す文字列とバージョン情報を示すヘッダ／論理画面のデータ構成／データの構成／ターミネータの構成／画像データの構成／インタレースGIFの技法／画像制御拡張子の構成／画像データの後ろにおかれる拡張情報の内容

ネットワークを介してやりとりする画像ファイル・フォーマット

第1-5章 PNG ……… 82

GIFフォーマットを置き換えることを最大の目的として作られた／PNGの普及と今後の問題点／従来の画像フォーマットとPNGの比較／PNGファイルの構成／PNG識別子のフォーマット／チャンクの基本フォーマットと命名規則／重要チャンクのパラメータ／従属チャンクの構成／LZ77方式を改良したDeflate／Inflate圧縮

Windows 標準のディジタル音声データ・フォーマット
第1-6章 WAVE 93
Microsoft 社のファイル・フォーマット RIFF のサウンド・データ／WAVE は長時間の音声データを扱うのは不向き／RIFF のフォーマットの構成／RIFF フォームの表記法／ウェーブフォーム・オーディオのデータ構成／WAVE のパラメータの計算／WAVE のデータの構成

ディジタル音声データの圧縮に使われる標準フォーマット
第1-7章 MP3 102
MPEG-1 と MP3／MP3 の普及と問題点／MPEG-1 のシステム仕様と MP3（MPEG-1 Audio Layer3）オーディオ仕様／MP3 のシステム仕様／圧縮技法の概要／MP3 データのフォーマット／MP3 における著作権保護の問題／MP3 の拡張フォーマットについて／MP3 の特許ライセンス

PC/AT 互換機のモデム・インターフェースや同期/非同期通信規格
第2-1章 シリアル・ポート 125
IBM 社の PC/AT 用のシリアル通信アダプタが始まり／シリアル・ポートと RS-232-C の違い／規格化されたシリアル・ポート EIA/TIA-574／マイクロソフトの PC 規格 PC 95 から PC 98 へ／非同期モデム，パソコンどうし，シリアル通信のための取り決め／非同期モデム・インターフェース／パソコンどうしの接続／シリアル通信コントローラ／8250/16450/16550A のレジスタ構成／シリアル・ポートの伝送速度／シリアル・ポートの電気的仕様

PC／AT 互換機のためのプリンタ・インターフェース規格
第2-2章 パラレル・ポート（IEEE 1284） 137
IBM 社の PC 用セントロニクス互換プリンタ・インターフェース/規格化されたパラレル・ポート IEEE 1284／PC 95 から PC 98 へ／パラレル・ポートとセントロニクス仕様の概要／セントロニクス仕様という規格／PC/AT のパラレル・ポート／IEEE 1284 規格の概要／IEEE 1284 のピン接続／互換モードのハンドシェイク／ネゴシエーションをしてモード変更をする／ニブル・モードのデータ転送／バイト・モードのデータ転送／ECP モードのデータ転送／EPP モードのデータ転送／電気的仕様

ユーザから見れば，そう簡単にはなくならない
第2-3章 PS/2 キーボード・インターフェース 160
AT 互換機のキーボード・インターフェース／AT 互換機のキーボードの変遷／キーボード・インターフェースの仕様／パソコンからキーボードへの送信／キーボードからパソコンへの送信／システムと 8042 のインターフェース

キーボード・インターフェースを拡張して作られた
第2-4章 PS/2 マウス・インターフェース 170
AT 互換機のマウス・インターフェース／キーボード・コントローラの空きポートを利用した PS/2 マウス／PS／2 マウスとシリアル・マウス

表紙デザイン/アイドマ・スタジオ　表紙イラスト/伊部喜久雄　本文イラスト/神崎真理子

はじめに

本特集の読み方

宮崎 仁

本特集は最近のパソコンやマルチメディア機器に関連するさまざまな規格をまとめたものです．1998年7月に発行したトランジスタ技術SPECIAL No. 63「特集 パソコン周辺インターフェースのすべて」，1999年7月に発行したトランジスタ技術SPECIAL No. 67「特集 パソコン周辺機器インターフェースII」の続編にあたります．

「特集 パソコン周辺インターフェースのすべて」では，Windowsパソコンの基本仕様とも言えるPC97，PC98から始まり，ISAやPCIなどシステム・バスの規格，USBやIEEE 1394など周辺機器バスの規格，PCカード，CardBus，ZVポートなどPCカード関連の規格，MIDIやNTSCなど民生用オーディオ/ビデオの規格，VGAやSVGAなどパソコン・グラフィックスの規格を取り上げました．

また，「特集 パソコン周辺機器インターフェースII」では，ハードディスクやCD-ROM，フラッシュ・メモリなどの大容量記憶装置をおもなターゲットとして，ATA（IDE），ATAPI，PCカードATA，各種の小型メモリ・カード，SCSIの規格をまとめて取り上げました．合わせて，グラフィック専用高速インターフェースのAGPを取り上げています．

この2回の特集では，おもにシステムやハードウェアに関する規格を紹介してきました．それによって，現在使われているwindowsパソコンがどのようなものであるか，かなり明らかになってきたと思います．

今回はちょっと趣を変えて，まず第1部として，マルチメディア・パソコンやインターネットの普及とともに重要性を増しているマルチメディア関係のデータ・フォーマットの規格を取り上げてみました．前半はJPEG，GIFなどの画像（静止画）データ，後半はWAVE，MP3などの音声データを紹介します．動画と音声を統合したMPEGなどの動画フォーマットもたいへん重要で興味深いものですが，残念ながらページ数の関係もあり，次回に回したいと思います．

さらに，特集の第2部として，ハードウェアに関する規格の中で，これまで取り上げることができなかったシリアル・ポート，パラレル・ポートなどのいわゆるレガシー・インターフェースを取り上げます．

とくに，パラレル・ポートについてはほかにあまり解説書が見当たらないことから，詳しく解説しました．

ディジタル画像フォーマット

ディジタル画像フォーマットは，大別するとラスタ・グラフィックス（いわゆるビット・マップ画像）とベクタ・グラフィックスに分けることができます．

一般的な画像フォーマットでは，このラスタ・グラフィックスがもっとも多く用いられています．本特集で取り上げたJPEG，TIFF，DIB（BMP），GIF，PNGは，いずれもこのラスタ・グラフィックスのフォーマットです．これらのフォーマットはそれぞれ異なりますが，画像をビット・マップで表していることは共通です．また，この中でPNGはごく最近（90年代後半）に作られたフォーマットですが，その他はいずれも80年代後半に骨格が作られた比較的古いフォーマットです．

最初のバージョンが世に出た年は，TIFFが1986年，GIFが1987年，DIBが1990年（Windows 3.0用），JPEGが1990年（最終ドラフト），そしてPNGが1996年となっています．すなわち，PNGを除けばいずれも80年代後半に作られ，10年以上にわたって使い続けられている実績があります．対応アプリケーションも豊富に揃っています．また，いずれも用途を限定しない汎用フォーマットであり，DIBを除けばいずれもマルチプラットフォームのデータ交換フォーマットを意識して作られています．

一方，ベクタ・グラフィックスはデータ量が小さく，拡大しても画像が荒れないなどの優れた特徴をもちます．ベクタ・グラフィックスはおもにDTPのページ記述や，CADデータなどといった特定の専門分野で使われており，そこではすでに長い実績があります．しかし，汎用の画像フォーマットとしてはまだあまり使われていないため，今回は取り上げるのを見送りました．

ただ，最近ではインターネット上でベクタ・グラフィックスを扱うための仕様（SVG）の標準化が進められており，将来的にはもっと普及していく可能性もあります．そのあたりの事情について少し解説します．

●ラスタ・グラフィックスとデータ圧縮

ラスタ・グラフィックスは，画像を細かく分割して点の集まりとし，各点（ドットあるいはピクセルと呼ぶ）に1ビットまたは複数ビットの情報を対応させたものです．点の集合にビットの集合を写像（マッピング）するということから，ビット・マップと呼ばれます．

パソコンの画面表示はこの方式であり，VRAM（グラフィック・メモリ）に書き込まれた1画面分のビット・マップ・データがCRTなどのディスプレイに表示されます．テキスト・データも，文字フォントを用いてビット・マップ・データに変換され，VRAM上でグラフィック・データと合成されます．もっとも簡単な形式の画像データは，このVRAMに書き込むビット・マップ・データをそのままファイル化したものです．ベタ形式などと呼ばれます．

ラスタ・グラフィックスでは，画像の分割は列（縦），行（横）の格子状に分割するのが普通です．画像全体をP列，Q行に分割した場合，全体のドット数は$P \times Q$になります．画像の物理的サイズを一定とすれば，分割数が大きいほど1ドットの物理的サイズは小さくなり，より滑らかで細部まで表現することが可能になります．そこで，この分割数を解像度と呼びます．

各ドットが1ビットなら，表示する/しないの2値データですから，モノクロの2値画像表示になります．線画と塗りつぶしだけの単純な画像や，文字の表示に適しています．ただし，白黒とは限らず，背景色と前景色の2色表示にすることは可能です．

また，各ドットにNビット（$N \geqq 2$）が割り当てられていれば，2^N種類の色（もしくは階調）を表現することができます．単純な画像でも，線や塗りつぶしにさまざまな色を用いることができるようになります．さらに，写真などの自然画像を表現できるようになります．ビット数(N)が多いほど色数を増やすことができ，表現力が向上します．

ラスタ・グラフィックスは原理が単純で簡単に処理できるのが利点ですが，一方でデータ量が大きくなってしまう欠点があります．たとえば，VGA相当の640×480ドットの解像度で，フル・カラー（ドットあたり24ビット＝3バイト）の画像を表示するとすれば，1画面分は640×480×3＝900Kバイトのデータ量になります．最近のパソコンの性能から見れば，これぐらいのデータ量は全然たいしたことはないように思えます．しかし，インターネットなどを通じて画像データをやり取りするとすれば，かりに64kビット/sで転送するとしても，

$$t = 640 \times 480 \times 24 / 64000 = 115.2 \text{ (s)}$$

すなわち，2分弱の時間がかかってしまいます．1画面の書き換えごとに2分も待たされるのでは，ユーザはたいへんなストレスを感じてしまいます．

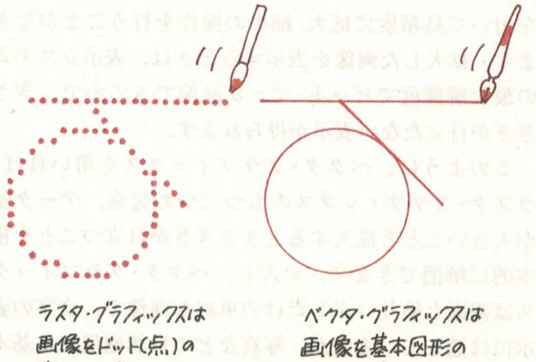
そのため，ラスタ・グラフィックスの画像フォーマットでは，データ圧縮の技術がたいへん重要になります．ほとんどの画像フォーマットは何らかのデータ圧縮手法を規格に取り入れています．JPEG，GIF，PNGでは，画像データはかならず決められた方法で圧縮，符号化して保存します．いずれも約1/2～1/10という高い圧縮効率が得られることで定評があります．インターネット上でWebサイトに画像を表示するための標準フォーマットとして，この三つが採用されています．

とくに，JPEGは視覚的に目立たない細部の情報を巧みに間引くことにより，ほかの圧縮方式では圧縮が難しい写真などの自然画像をきわめて高効率に圧縮できるのが特徴です．そのかわり，一度圧縮したら完全には元に戻せない非可逆圧縮と呼ばれる方式になっています．なお，JPEGの規格には可逆圧縮の方式も含まれていますが，実用的にはほとんど非可逆圧縮だけが使われています．

DIB（BMP）とTIFFは，画像データを圧縮して保存することもできますが，無圧縮のままで使われることが多いフォーマットです．また，TIFFでは複数の圧縮方式がサポートされていて，その中から用途に合わせて選択できます．

●ベクタ・グラフィックス

ラスタ・グラフィックスにはもう一つ基本的な欠点があります．ラスタ・グラフィックスでは，画像を点（ドット）の集まりとして扱います．ドットの物理的



《ラスタ・グラフィックスとベクタ・グラフィックスの違い》

サイズが十分に小さくないと，ぎざぎざ（ジャギー）が目立つ画像になってしまいます．最近のプリンタは解像度が十分に高く，ドットの物理的サイズもかなり小さいので，ぎざぎざが気にならないきれいな画像を見ることができます．ディスプレイの解像度はそれより1桁落ちますが，まあまあ満足できるレベルと言って良いでしょう．

しかし，小さい画像を拡大したときは話が別です．ビット・マップ画像を拡大して表示するには，ドット構成はそのままでドット・サイズが拡大されるため，拡大すればするほど，ぎざぎざが目立つ画像になります．一方，画像を縮小するときはドットを間引かなければならないので，細部の情報が失われます．

そこで，ラスタ・グラフィックスとはまったく異なった画像の表現方法として，ベクタ・グラフィックスが使われています．

ベクタ・グラフィックスは，画像を点の集まりとして表すかわりに，直線，円弧などの基本図形の集まりとして表します．そして，各基本図形は直線なら（始点，終点）あるいは（始点，方向，長さ），円弧なら（始点，終点，中心点）あるいは（始点，中心点，中心角）といったようなパラメータの組み合わせで表すことができます．さらに，単純な基本図形だけでなく，ベジェ曲線などの数式（関数）を用いて曲線を表すこともできます．

これらのデータはディスプレイやプリンタで直接表示できないので，ベクタ・グラフィックスを表示するときは，かならず最終的にビット・マップに展開してから表示を行います．

画像を点の集まりとして表すラスタ・グラフィックスにくらべて，少数のパラメータだけで表すことができるので，データ量はずっと小さくなります．さらに，表示システムの解像度とは無関係に，パラメータだけを用いて高精度に拡大/縮小の操作を行うことができます．拡大した画像を表示するときは，表示システムの最大解像度でビット・マップ展開できるので，ぎざぎざが目立たない表示が得られます．

このように，ベクタ・グラフィックスを用いれば，ラスタ・グラフィックスのもつ二つの欠点，データ量が大きいことと拡大するとぎざぎざが目立つことを根本的に解消できます．ただし，ベクタ・グラフィックスは線画と塗りつぶしだけの単純な画像や，文字の表示には適していますが，写真などの自然画像には基本図形の集まりで表すのが困難なものもあります．

ベクタ・グラフィックスは優れた特徴ももっていますが，最終的にビット・マップに展開する手間がかかるので，これまで汎用の画像フォーマットとしてはあまり使われず，ベクタ・グラフィックスの特徴を生かせる特定の用途で使われてきました．古くから知られているものとしては，CAD データの交換フォーマットである DXF（Autodesk 社），DTP のページ記述に使われる PostScript（Adobe 社）などがあります．また，Macintosh の標準画像フォーマットの PICT は，画像表示ツールの QuickDraw に対する描画命令をファイル化したもので，ベクタ画像とビット・マップ画像の両方を扱うことができます．

また，文字フォントでもビット・マップ・フォントは拡大するとぎざぎざが目立つことから，ベクタ・グラフィックスと似た考え方によるアウトライン・フォント（スケーラブル・フォント）が使われるようになりました．現在 Macintosh や Windows で標準的に使われている TrueType フォントは，アウトライン・フォントの代表的なものです．

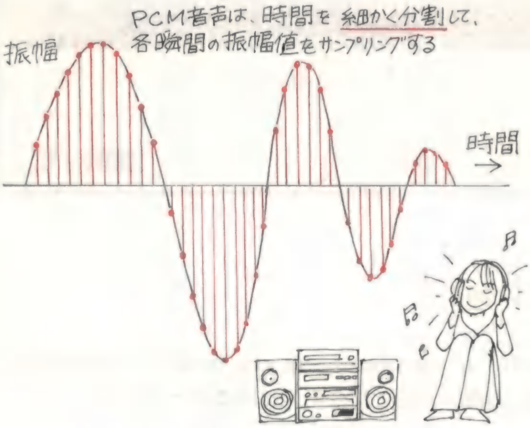
● 汎用ベクタ・グラフィックスの標準化

最近では，インターネットでもデータ量の小さいベクタ・グラフィックスを利用して Web サイトの表示を高速化しようという動きがあります．その代表的なものとして，PGML（Precision Graphics Markup Language）と VML（Vector Markup Language），それらを統合した SVG（Scalable Vector Graphics）があります．いずれも，文法的には XML（eXtensible Markup Language）ベースのページ記述言語であり，ベクタ・グラフィックスの取り扱いは PostScript をベースにしています．

PGML と VML は内容的には類似したものであり，業界の主導権争いから二つに分かれたものと言えるでしょう．PGML は Adobe 社が中心となって仕様を定め，Netscape 社，Sun Microsystem 社などが支持していました．VML は Autodesk 社，HP 社，Macromedia 社，Microsoft 社，Visio 社が共同で仕様を定めたものです．

インターネット関連技術の標準化を行っている W3C では，PGML と VML を比較検討した結果，両者の特徴を取り入れた新しい仕様として SVG を提案しました．各メーカも SVG に賛同しており，今後は SVG がベクタ・グラフィックスの中心となっていく見通しです．

しかし，SVG は 1999 年に規格案が公開されたものの，まだ標準化作業の段階であり，実際に普及を始めるのはかなり先になりそうです．現在のところ，Adobe 社では SVG 対応アプリケーションを積極的に



《情報のディジタル化＝サンプリング》

製品化していますが，他メーカはまだあまり積極的とは言えないようです．

ディジタル音声フォーマット

動画や音声のように時間的に連続した情報をディジタル化するには，まず情報を時間的に細かく分割して，各瞬間ごとの情報の列に置き換えることが必要です．これをサンプリングと呼びます．とくに音声の場合は，元の音声がもつ周波数成分を失わないように，十分に細かく時間を分ける必要があります．そのためには，帯域幅の少なくとも2倍以上のサンプリング・レートが必要です．

もっとも簡単な形式の音声データは，各瞬間における音声信号の振幅をサンプリングして，ディジタル・データの列に置き換えたものです．この方式はPCM (Pulse Code Modulation) と呼ばれるもので，古くは交換機などの電話システムに用いるために研究や実用化が進められてきました．その後，ハイファイ・オーディオにも取り入れられるようになり，ディジタル録音機やCD-DAのようなディジタル・オーディオ・ディスクに使われています．

PCM音声は，画像データにおけるビット・マップ画像に相当するものと言えるでしょう．ビット・マップ画像は画像を平面的に細分化してドットの集まりで表し，各ドットの明るさ（階調）をNビットのディジタル値に置き換えます．PCM音声は音声を時間的に細分化してサンプリングし，各瞬間の音声信号の振幅をNビットのディジタル値に置き換えます．

そのため，PCM音声もビット・マップ画像と同様にデータ量がきわめて大きいという欠点をもちます．

電話システムでは，もともとアナログ音声の帯域幅が3.4 kHzに制限されており，これをカバーするために8 kHz（3.4 kHzの2倍よりやや上）で音声信号をサンプリングしています．振幅方向の分解能は8ビットです．

動画や音声では1秒あたりのデータ量をビット・レート（ビット/s）で表しますが，電話のPCM音声では8k× 8＝64 kビット/sになります．ISDNの基本伝送レートは，このPCM音声を伝送できるように64 kビット/sに選ばれています．

ハイファイ・オーディオでは，音楽を歪みのない忠実な音で再現するために，少なくとも15 kHz，できれば20 kHzの帯域幅が必要だと言われています．そこで，CD-DA（音楽CD）では44.1kHz（20kHzの2倍よりやや上）で音声信号をサンプリングしています．また，振幅方向の分解能もCD-DAでは16ビットに拡大されています．

さらに，L, Rの2チャネルそれぞれを同じレートでサンプリングしますから，ビット・レートは44.1 k×16×2 ≒ 1.4 Mビット/sと大きくなります．インターネットでリアルタイムに伝送したり，ハードディスクやCD-ROMなどのメディアに蓄積するにはデータ量が大きく，容易には利用できません．

そこで，PCM音声を高効率に圧縮する方法が工夫されてきました．とくに，聴感上わかりにくいように情報を巧みに間引くことにより，高い圧縮率を得る方法がいくつも工夫されています．そのかわり，一度圧縮したら完全には元に戻せません．画像データにおけるJPEGに相当する非可逆圧縮になっています．

その代表的なものが1998年頃から世界的に爆発的な流行を見せたMP3（MPEG-1 Audio Layer3）です．MP3はCD並みの音質を保ちながら，ビット・レートを約1/11の128kビット/sに圧縮できると言われています．

ただし，MP3は著作権保護のための有効な手段が組み込まれていないため，CD-DAなど市販の音楽データの違法コピーに利用されるなどの問題点がずっと指摘されてきました．

最近では，さらに高効率の圧縮が可能で，かつ著作権保護と組み合わせたさまざまな方式の製品化が始まっています．それらの中からMP3に代わる次世代のデファクト・スタンダードが出てくるかどうか，まだ見通しは明らかではありません．

本特集では，PCM音声フォーマットとして広く用いられているWAVEと，MP3を取り上げて詳しく解説します．

第1-1章　もっとも広く使われているディジタル画像の圧縮/符号化方式

JPEG

宮崎　仁

〈名称〉

Joint Photographic Experts Group

〈規格名〉

Digital compression and coding of continuous-tone still images: requirements and guidelines（連続階調静止画像の圧縮と符合化：要求条件とガイドライン）

〈発行日〉

1992年9月（ITU-T勧告T. 81）

1994年2月（ISO/IEC 10918-1）

〈発行者〉

ITU-T

ISOおよびIEC

〈情報入手先〉

http://www.jpeg.org/（JPEG/JBIGホームページ）

http://www.itu.ch/（ITU-T）

http://www.iso.ch/（ISO）

http://www.iec.ch/（IEC）

http://www.c-cube.com/（C-Cube Microsystems）

〈関連規格〉

ITU-T T. 80（ディジタル画像圧縮の基本原則: JPEGとJBIGに共通の導入部）

ITU-T T. 83（JPEG適合性試験）

ISO/IEC 10918-2（JPEG適合性試験）

ITU-T T. 84（JPEG拡張: SPIFF）

ISO/IEC 10918-3（JPEG拡張: SPIFF）

ITU-T T. 86（JPEGプロファイル，SPIFFプロファイル，SPIFF色空間，APPnマーカ，SPIFF圧縮タイプの登録）

ISO/IEC 10918-4（JPEGプロファイル，SPIFFプロファイル，SPIFF色空間，APPnマーカ，SPIFF圧縮タイプの登録）

JFIF Ver. 1.02（C-Cube社が提唱する共通フォーマット仕様）

ディジタル画像データを圧縮する必要性から生まれた規格

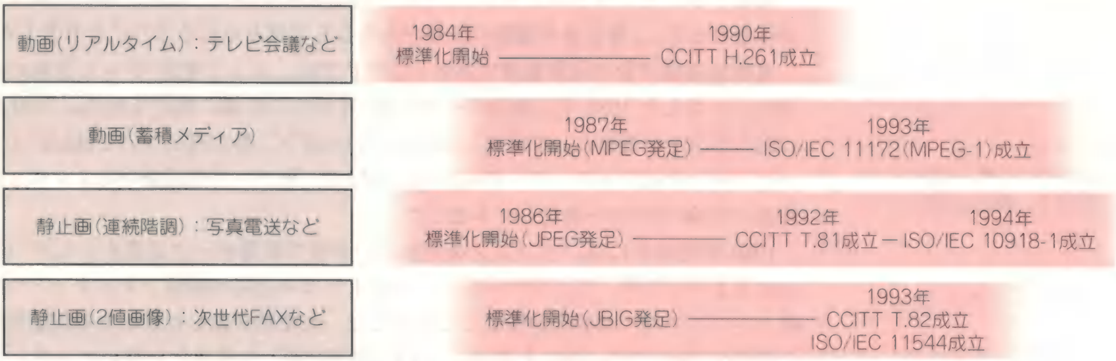
JPEG（ジェイペグ）(Joint Photographic Experts Group）は，静止画データの圧縮/符号化方式の国際標準を検討，作成するために1986年に設置された非公式の専門家グループの名称です．国際的な標準化機関であるISO/IECとCCITT（現在はITU-T）の合同プロジェクトとして発足したことから，Jointという名称が付けられました（図1-1）．

また，このグループによって作成された圧縮/符号化方式や国際標準もJPEGと呼ばれています．ここでは，とくにまぎらわしい場合には，専門家グループを

Jointという名称

最近では，ISO, IEC, ITUなど複数の団体が協調しながら標準化を進めるようになったことから，Jなんとかという名称の団体や規格が増えてきた．決して，JapanのJではない．

〈図 1-1〉画像圧縮/符号化規格の変遷



JPEG 委員会，圧縮/符号化方式を JPEG 方式，国際標準を JPEG 規格と呼んで区別することにします．

JPEG 委員会はカラーおよびモノクロの自然画像（連続階調）データの圧縮/符号化方式を検討し，1990 年に規格の最終案を完成しました．CCITT では 1992 年 9 月に勧告 T.81 として承認され，ISO/IEC では 1994 年 2 月に標準 10918-1 として承認されました．規格が成立した時期は多少ずれましたが，内容は同等です．また，CCITT T.81，ISO/IEC 10918-1 のいずれにも，最終的な国際標準は CCITT/ISO/IEC の共通テキストに従うべきであると書かれています．

JPEG 方式はフルカラーの自然画像データを効率よく圧縮できることから広く普及し，現在もっとも広く使われている画像の圧縮/符号化方式といえるでしょう．とくに，インターネットで静止画データを扱うためのデファクト・スタンダードとなっています．また，多くのディジタル・スチルカメラでは，画像データの圧縮/記録に JPEG 方式を使っています．

また，JPEG で採用された 2 次元 DCT（Discrete Cosine Transform）方式は，MPEG などの動画データ符号化でも基本方式として使われています．

ただし，JPEG は単純な線画やアニメーションなどの 2 値画像には不向きです．2 値画像を JPEG で圧縮/符号化すると，圧縮の効率が低いだけでなく，**画質も低下**します．

> **画質も低下**
> 一色で塗りつぶされた部分を JPEG で圧縮すると，8×8 ピクセルのブロック状のノイズが現れることが知られている．また，直線やくっきりした輪郭はぼやけてくる．

JPEG のもう一つの難点は，JPEG 規格（T. 81/10918-1）だけではファイル・フォーマットの規定が不十分なことです．JPEG 準拠のアプリケーションが生成した JPEG ファイルを，ほかの JPEG 準拠のアプリケーションが表示できない場合もあります．このため，JPEG ファイルに最小限の互換性をもたせるための共通仕様として，JFIF（JPEG File Interchange Format）が利用されています．JFIF 仕様は，JPEG 委員会や ITU-T，ISO/IEC ではなく，画像処理 LSI の有力ベンダである C-Cube（シーキューブ）社が作ったものです．

現在では，一般に JPEG ファイルと呼ばれているものは，大部分がこの JFIF 仕様に準拠していると言われています．

JPEG 規格の成り立ちと規格化の過程

● 動画/静止画の画像データ圧縮/符号化規格 H. 261 と JPEG，JBIG

CCITT では早くから画像データの圧縮/符号化方式の標準化に取り組んできました．

1980年代にはモデムやISDNなど電話回線を使ったデータ通信の技術が急速に発展しました．当初はテキスト・データの伝送が主体でしたが，通信の高速化が進むとともに，静止画や動画の伝送があちこちで試みられるようになりました．

電話回線提供者の国際組織であるCCITTでは，テレビ電話/テレビ会議の実用化をめざして1984年に動画データ圧縮/符号化の仕様の検討を始め，1990年に勧告H.261が規格として成立しました．いっぽう，静止画についてはビデオテックスや次世代ファクシミリの実用化に向けて，カラー自然画像やモノクロ2値画像の圧縮/符号化を検討してきました．

ビデオテックス
電気紙芝居，と呼ぶのがもっとも近いだろう．最近は話題にもならない．

1980年代後半には，パソコンなど幅広い分野で画像データ処理が少しずつ行われるようになり，アプリケーションを限定しない汎用の画像フォーマットや圧縮アルゴリズムへの要求が強まってきました．産業界全般を対象とする国際標準化機関のISOでは，IECと共同してIT（情報技術）の標準化を進めていましたが，そこでも画像の圧縮/符号化が検討されるようになりました．

そこで，ISO/IECとCCITTが協同して国際標準化を進めることになり，JPEG（Joint Photographic Experts Group），JBIG（Joint Bi-level Image Group）の二つの非公式の合同専門家グループが作られました．JPEG委員会はおもにフルカラーの自然画像データの圧縮/符号化を担当し，国際標準はCCITT勧告T.81（1992）とISO/IEC標準10918-1（1994）として成立しました．また，JBIG委員会はJPEG委員会から分化したもので，おもにモノクロ2値画像データの圧縮/符号化を担当しました．JBIGの国際標準はCCITT勧告T.82（1993）とISO/IEC標準11544（1993）として成立しました．

非公式の合同専門家グループ
よりよい規格を作るためには，メンバーが立場や肩書きにこだわらず自由に討議できる環境を作ることが必要だということだろう．

ソフト間でJPEGファイルを交換するための共通フォーマットJFIF

JPEG規格は静止画データの圧縮/符号化のアルゴリズムを規定し，また符号化データの基本的なフォーマットを規定しています．しかし，復号化に必要な付加情報の部分はアプリケーションの自由に任された部分が多く，共通フォーマットの仕様を明確に規定できませんでした．そのため，パソコン環境やアプリケーションごとに異なるJPEGファイルが出現し，互換性が失われるという問題がありました．このことは規格案の作成中からすでに問題となっていましたが，結局解決されないままT.81/10918-1の規格ができました．

結局解決されないまま
当時の関係者の発言などを読むと，ISO/IECのほかの専門委員会からいろいろな圧力がかかり，JPEGではデータ交換フォーマットを扱うべきでない（圧縮/符号化に専念すべし）という方向に進んでいったようである

そこで，有力な画像処理LSIメーカであるC-Cube社が，アプリケーション間でJPEGファイルを交換するための最小限の共通フォーマットとしてJFIF（Jpeg File Interchange Format）を提唱しました．JFIFはJPEGの規格に準拠しており，ファイル交換に必要な最小限の付加情報を記述するフォーマットを共通化したものです．

T.81の成立に合わせてJFIF Ver.1.02の仕様書（1992年9月）が発行されています．そのpdf版は，C-Cube社のWebサイトでダウンロードできます．

JFIFはJPEG委員会やITU-T，ISO/IECとは直接関係のない私的な規格ですが，デファクト・スタンダードとして広く普及しました．現在，一般に流通しているJPEGファイルや，エンコーダ/デコーダのハードウェア/ソフトウェアの大部分は，このJFIF仕様に準拠していると言われています．JFIF準拠と明記していない単なるJPEGファイルでも，実際にはJFIFに準拠している確率が高いでしょう．とくに，インターネットではW3C（World Wide Web Consortium）がJFIF仕様を採用していることから，JFIF準拠のJPEGファイル（JPEG JFIF）が標準フォーマットの一つとして扱われています．

W3C
W3CのWebサイトには，JPEG JFIFのページがある．JFIFの仕様書もダウンロードできる．

JPEGのパート2，パート3，…

ITU-T T.81とISO/IEC10918は静止画データの圧縮/符号化に関する要求条件とガイドラインを規定したもので，JPEG規格のパート1に相当します．それに続いて，JPEG委員会ではJPEG規格のパート2として**適合性試験の規格**を作成し，ITU-T T.83（1994）とISO/IEC 10918-2（1995）となりました．この規格は，圧縮データのフォーマットやエンコーダ（符号器），デコーダ（復号器）が仕様を満たしているかどうかを判定するためのものです．

適合性試験の規格
ハードウェア，ソフトウェアの製品がちゃんと規格に適合しているかどうかを判定するための試験方法を定めたもの．

さらに，共通フォーマットの問題を解決するために，JPEG委員会ではファイル交換のための共通フォーマットSPIFF（Still Photographic Interchange File Format）の仕様などを定めた拡張JPEGの規格を作成しました．これはJPEG規格のパート3であり，ITU-T T.84（1996）とISO/IEC10918-3（1997）として成立しました．

SPIFFはJFIFに対して上位互換として作られており，JFIFを置き換えることをねらったものです．しかし，今のところJFIFのほうがデファクト・スタンダードとして一般的に使われているようです．

また，JPEG規格のパート4として，JPEGプロファイルやSPIFFプロファイルなどの登録の規格が作られています．ITU-T T.86（1998）とISO/IEC10918-4（1999）として成立しました．

あらゆる静止画データを変換できるフォーマットJPEG2000

現在，JPEG委員会とJBIG委員会は共同して，あらゆる静止画データを扱える次世代の圧縮/符号化方式として，**JPEG2000**の標準化を進めています．

JPEG2000
あまりに壮大なプロジェクトなので，2000年中にはでき上がりそうにない．

これは，2値画像，グレー・スケール画像，カラー画像，マルチコンポーネント画像，自然画像，文字画像などのデータを統一的に扱うことを目標としています．想定されているアプリケーションも，ファクシミリ，スキャナ，印刷，ディジタル写真，医療，リモート・センシング，モバイル，電子商取り引きなどきわめて幅広くなっています．従来のJPEGとJBIGを統合して，さらに大幅に拡張した規格と言えるでしょう．

また，従来のJPEGにはなかったビット・エラー対策や，セキュリティ対策もJPEG2000には盛り込まれています．

ただし，このJPEG2000が成立したとしても，すぐに従来のJPEGやJBIGに取って代わるというわけではなさそうです．

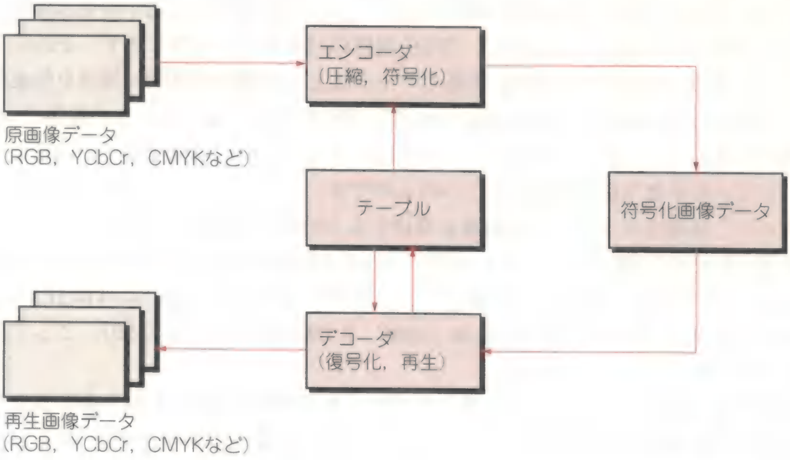
JPEG －連続階調の静止画データを圧縮/符号化する方式

JPEG規格では，連続階調の静止画データを圧縮/符号化し，またそれを復号化/再生するための方式を定めています．特定のプラットフォームやアプリケーションに限定されない汎用の規格です．また，JPEG規格には複数の圧縮/符号化の方式が含まれており，必要に応じて最適な方式を選択できます．その中でももっとも一般的に使われる方式は，ベースライン・プロセス（あるいはベースラインJPEG）と呼ばれるものです．

原画像データを圧縮して符号化画像データを生成する機能をエンコーダ（符号器），符号化画像データを復号化して再生画像データを生成する機能をデコーダ（復号器）と呼んでいます（**図1-2**）．エンコーダ/デコーダの実現方法は任意であり，ソフトウェアでもハードウェアでも実現できます．

JPEGではかならず圧縮を行います．非圧縮のモードはありません．圧縮には

〈図 1-2〉エンコーダとデコーダ



可逆圧縮は圧縮率が低い
一般論としても，情報を切り捨てるほどデータは小さくなるので，情報を切り捨てない可逆圧縮はそれほど圧縮率を上げられない．ただ，JPEGの可逆圧縮方式は，ほかの可逆圧縮方式とくらべてもあまりできがよくないと言われている．

非可逆圧縮（Lossy Compression）と可逆圧縮（Lossress Compression）の二つの方式があり，可逆圧縮なら再生画像データは原画像データと一致します．しかし，**可逆圧縮は圧縮率が低い**ため，実際にはあまり使われていません．

そのまま画像データは原画像データと一致しませんが,圧縮率を高くできます．この非可逆圧縮には，DCT（離散コサイン変換）方式が使われます．

ベースライン・プロセスは，非可逆圧縮クラス，シーケンシャル符号化モード，ハフマン符号化手順という方式です．

画像データの形式

● 扱える画像データ形式

JPEGでは，1個の静止画データを圧縮/符号化して，1個のファイルを生成します．複数の画像を1個のファイルにまとめる機能はありません．

ただし，1個の原画像から解像度の異なる複数の画像データを作り，それを1個のファイルにまとめる方法はあります．たとえば，**サムネール**などの低解像度画像と元の画像を1個のファイルにまとめることは可能です．しかし，それには階層モードと呼ばれる特別なモードを使う必要があるため，実際にはあまり使われていません．

サムネール
大きな画像のダウンロードや表示には時間がかかるので，概略だけわかる程度の小さい画像がよく使われる．サムネールは親指の爪に描く小さい絵のこと．

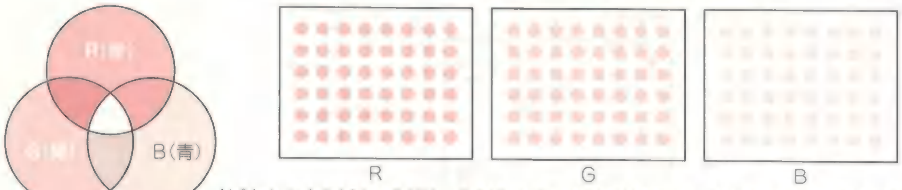
原画像はラスタ・グラフィック（ビット・マップ）データであれば，モノクロ（グレー・スケール）画像でも，RGBやYCbCr, CMYKなどのカラー画像でも同じように扱えます．モノクロでもカラーでも，自然画像（連続階調画像）が対象であり，線画や文字などの2値画像は対象としません．また，カラー画像はフルカラー（コンポーネント・カラー）が対象であり，**パレット方式のカラー画像**はそのままでは扱えません．

パレット方式のカラー画像
昔のVGAの16色表示や，初期のWindowsの標準の256色表示に使われていた．あらかじめ，16色なり256色を作っておいて，呼び出して使う方式．パレットの上で調合しておいた絵の具で絵を描くのと似ているので，パレット・カラーと呼ぶ．

ディスプレイ装置でおもに使われるRGB方式はR（赤），G（緑），B（青）の三つのコンポーネント，YCbCr（YUV4：2：2）方式はY（輝度），Cb（色差B-Y），Cr（色差R-Y）の三つのコンポーネントからなります．また，プリンタや印刷機などに使われるCMYK方式はC（シアン），M（マゼンタ），Y（イエロー），K（ブラック）の四つのコンポーネントからなります（**図 1-3**）．

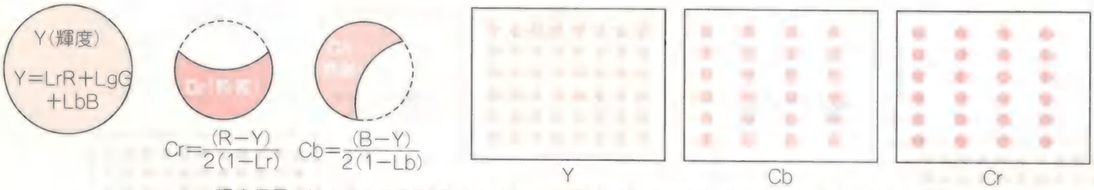
JPEGでは各コンポーネントを別々に圧縮/符号化しますから，画像の中から

〈図1-3〉コンポーネント・カラー表示



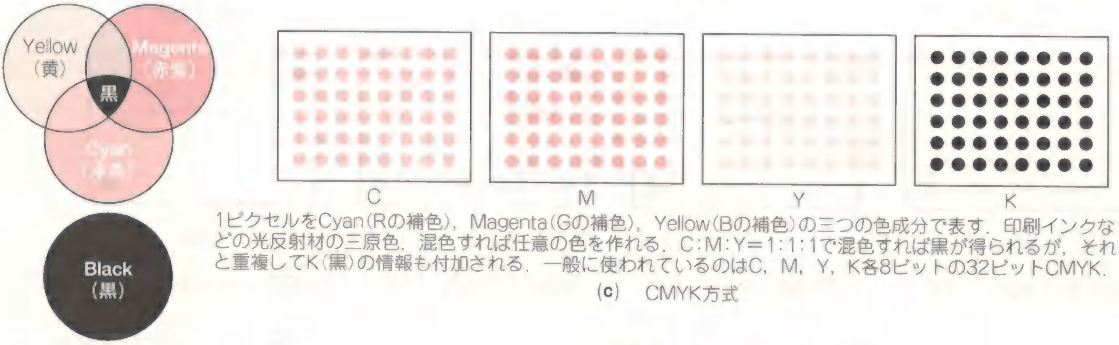
1ピクセルをR(赤)，G(緑)，B(青)の三つの色成分で表す．光の三原色(R，G，B)を各サンプル値の比率で混色すれば任意の色を作れる．一般に使われているのはR，G，B各8ビット(256階調)の24ビットRGB．

(a)　RGB方式



輝度信号Y(R+G+B成分)と二つの色差信号Cb(R−Y成分)，Cb(B−Y成分)で任意の色を作れる．Lr，Lg，LbはYCbCrとRGBの間の変換係数．色差信号の解像度を輝度信号の1/2に落としてデータ量を減らしている．一般に使われているのはY，Cb，Cr各8ビットの方式．データ量は16ビット/ピクセル相当．YUV4:2:2方式は，厳密にはコンポジット信号向けの方式で，これと若干異なる．

(b)　YCbCr方式



1ピクセルをCyan(Rの補色)，Magenta(Gの補色)，Yellow(Bの補色)の三つの色成分で表す．印刷インクなどの光反射材の三原色．混色すれば任意の色を作れる．C:M:Y=1:1:1で混色すれば黒が得られるが，それと重複してK(黒)の情報も付加される．一般に使われているのはC，M，Y，K各8ビットの32ビットCMYK．

(c)　CMYK方式

特定の色を抽出したり，それを他の色に変更したりすることは本来苦手です．特定の色を透明にする透過GIFのような機能はもっていません．

透過GIF
画像のある部分の色を透明にして，背景色が見えるようにする手法．背景画の中にロゴやキャラクタを貼り付けることができる．それを少しずつ動かせば，アニメーションになる．

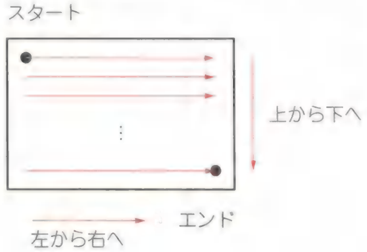
● 画像の解像度の制限と階調のビット数

JPEGでは，画像の解像度（ピクセル数）の上限は最大64 K × 64 Kピクセルと大きく，実用上は無制限と見なせます．その範囲では解像度は任意です．ただし，非可逆圧縮では原画像データを8 × 8ピクセルのブロックに分割して扱うため，解像度が8ピクセルの倍数でないときは8ピクセルの倍数になるようにデータを付加して圧縮/符号化します．付加した分のデータは，アプリケーション側で取り除いて表示できます．

画像の階調（各コンポーネントのビット数）は方式によって異なりますが，ベースライン・プロセスではコンポーネントあたり8ビットと決まっています．その他のプロセスでは最大12ビットまたは16ビットに拡張できます．階調は符号なし整数で定義されます．すなわち，8ビットなら0～65535の範囲です．

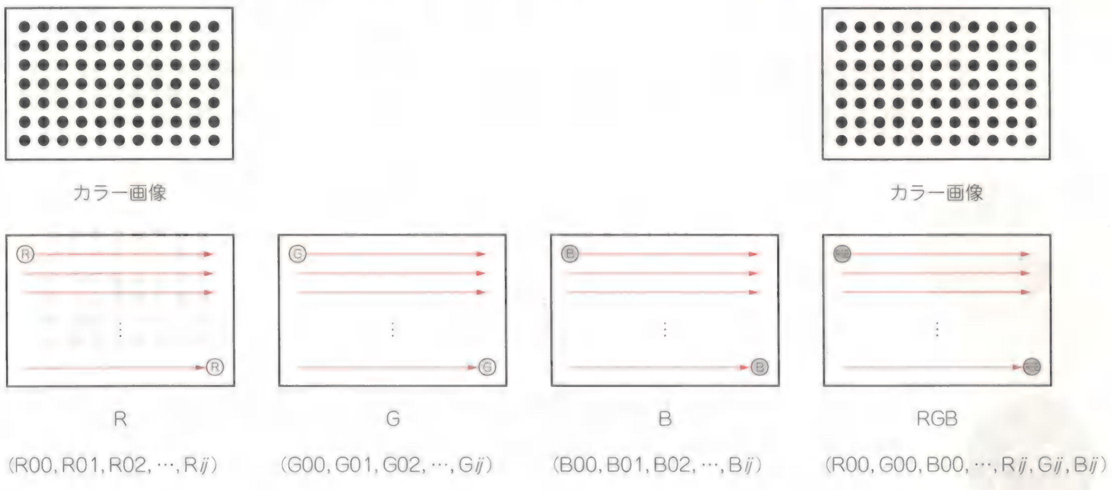
原画像の階調は任意ですが，原画像データの階調が8ビット未満の場合，非可逆圧縮ではデータを8ビットに拡張してから圧縮/符号化します．可逆圧縮は，2～16ビットの階調をそのまま扱うことができます．

〈図 1-4〉トップダウン・スキャン



DIB(p.57参照)などごく一部の例外を除いて，トップダウン・スキャンが採用されている．

〈図 1-5〉ノンインタリーブ・スキャンとインタリーブ・スキャン



(a) ノンインタリーブ・スキャン　(b) インタリーブ・スキャン

医用機器などの特定のアプリケーション

一般に，12 ビット（4096 階調）のモノクロ画像がよく使われるようである．

一般的なモノクロ画像や CMYK, RGB, YCbCr などのフルカラー画像では，各コンポーネントの階調は 8 ビットなので，実用上はベースライン・プロセスの仕様で十分です．医用機器など特定のアプリケーションでは，8 ビットを超える高階調のモノクロ画像を扱いたいという要求があります．

画像のスキャン

1 画面分の画像を読み取ったり，表示することをスキャン（走査）と呼びます．JPEG では，画像の左上隅の点からスキャンを開始します．左から右に水平方向のスキャンを行い，それを上から下に向けて繰り返します (top-down scan)．画像の右下隅の点に達したら 1 回のスキャンが終了します (図 1-4)．

非可逆圧縮では，8×8 ピクセルのブロックを単位としてスキャンを行います．すなわち，画像の左上隅の 8×8 ブロックからスキャンを開始し，右下隅の 8×8 ブロックでスキャンを終了します．

可逆圧縮では 1 ピクセルずつスキャンを行います．

モノクロ画像は 1 個のコンポーネントですから，1 回のスキャンですべての画像データを処理できます．それに対して，カラー画像のように複数のコンポーネ

ントから構成される画像では，二つのスキャン方法があります（**図1-5**）．

一つは，各コンポーネントを個別にスキャンする方法です．たとえば，RGBカラー画像であれば，まずRについて左上から右下までスキャンし，次にGについて左上から右下までスキャンし，次にBについて左上から右下までスキャンします．これを**ノンインタリーブ・スキャン**と呼びます．*N* 個のコンポーネントからなる画像をノンインタリーブ・スキャンする場合，すべての画像データを処理するには *N* 回のスキャンが必要です．

もう一つは，各コンポーネントの同じピクセル（または8×8ブロック）を次々にスキャンし，それを左上から右下まで繰り返す方法です．たとえば，RGBカラー画像であれば，まず左上のピクセル（または8×8ブロック）をR，G，Bの順でスキャンし，次のピクセル（または8×8ブロック）をR，G，Bの順でスキャンし，その次のピクセル（または8×8ブロック）をR，G，Bの順でスキャンし，…，というようにスキャンします．これを**インタリーブ・スキャン**と呼びます．インタリーブ・スキャンの場合は，コンポーネントが何個あっても，1回のスキャンですべての画像データを処理できます．

画像形式によっては，コンポーネントによってピクセル数が異なるものがあります．たとえば，CCIR勧告601として知られるYCbCrフォーマットでは，輝度Yの水平解像度に対して，色差Cb，Crの水平解像度はそれぞれ1/2となっています．この場合，Yを2ピクセル（または2ブロック），Cbを1ピクセル（または1ブロック），Crを1ピクセル（または1ブロック），…，というように，解像度の比に応じたピクセル数（またはブロック数）ずつスキャンすれば，インタリーブ・スキャンが可能です．

ノンインタリーブ・スキャン
昔のパソコンは，R，G，Bの各色ごとにそれぞれ専用のVRAM（カラー・プレーン）をもち，それを合成して表示色を作っていた．それに似たやり方．現在でも，印刷版下などは各色（印刷の場合はC，M，Y，K）に分解した色版を使うので，ノンインターリーブ・スキャンが適している．

インタリーブ・スキャン
最近のパソコンは，1ピクセル分のRGBデータ（たとえば24ビット）をひとまとめに扱うことが多いようである．

JPEG規格書の構成

JPEGの規格書の本文には，圧縮/符号化方式の概要，エンコーダに対する要求条件，デコーダに対する要求条件が記述されています．一方，圧縮/符号化の具体的な方法や，画像ファイルの基本フォーマットなどは巻末の**付属書**A～Jに規定されています．これらの付属書は標準内規定として扱われています．その後にもさらに付属書K～Mが続きますが，これらは参考資料です．

ISO/IEC10918-1の構成を**表1-1**に示します．ITU-T T.81は多少順番が入れ替わっていますが，内容は同等です．

付属書
最近の規格書を見ると，本文は規格の位置付けや用語の定義に終始し，具体的な内容はすべて付属書に書かれている場合もある．付属書と侮ってはいけない．

圧縮/符号化方式にはどういうものがあるか

圧縮/符号化方式の分類

JPEGでは複数の圧縮/符号化方式を採用し，必要に応じて使い分けられるようになっています．圧縮/符号化方式を分類，整理するために，二つの符号化クラス，四つの符号化モード，二つの符号化順序，二つの符号化手順を規定しています（**表1-2**）．

さらに，それらを四つの代表的な符号化プロセスに分類しています（**表1-3**）．

非可逆圧縮クラスと可逆圧縮クラス

非可逆圧縮（DCT利用型）クラスは，圧縮時に原画像のもつ情報の一部が失われます．復号化によって完全に原画像に戻すことはできず，画質は低下しますが，そのかわり高い圧縮率が得られます．

〈表 1-1〉ISO/IEC10918-1 規格書の構成

本文		
1.	規格の範囲	
2.	標準参照文献	
3.	定義，略号および記号	
4.	一般的事項	
5.	交換フォーマットの要求条件	
6.	エンコーダの要求条件	
7.	デコーダの要求条件	
付属書		
A	数学的定義	標準内規定
B	圧縮データ・フォーマット	標準内規定
C	ハフマン・テーブル指定	標準内規定
D	算術符号化	標準内規定
E	エンコーダとデコーダの制御手順	標準内規定
F	DCT 利用型シーケンシャル・モード	標準内規定
G	DCT 利用型プログレッシブ・モード	標準内規定
H	可逆モード	標準内規定
J	ハイアラーキカル・モード	標準内規定
K	例とガイドライン	参考
L	特許	参考
M	参考文献	参考

ITU-T T.81 は本文の 3 章と 4 章の順序が逆

〈表 1-2〉JPEG に規定された方式

符号化クラス	
非可逆圧縮	Lossy Compression
可逆圧縮	Lossress Compression

符号化モード	
DCT 利用型シーケンシャル	Sequential DCT-based
DCT 利用型プログレッシブ	Progressive DCT-based
可逆	Lossress
階層	Hierarchical

符号化順序	
インタリーブ	Interleaved
ノンインタリーブ	Non-interleaved

符号化手順	
ハフマン符号化	Huffman Coding
算術符号化	Arithmetic Coding

自然画像データの特性を利用して
自然画像データには，低周波から高周波までさまざまな周波数の成分がランダムに含まれている．それらの中から少しずつ情報を間引いても視覚的に目立ちにくいと考えられる．それに対して，同じ色での塗りつぶし，一定パターンの繰り返し，白と黒が交互に現れるモノクロ 2 値画像などは，特定の周波数成分だけに偏っている．そこから情報を間引いていくとどうしても目立ってしまう．

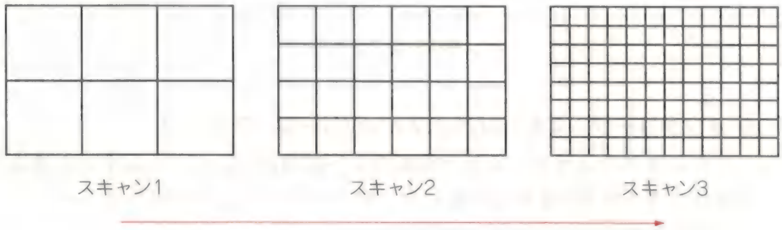
画質が低下するといっても，**自然画像データの特性を利用して**，視覚的になるべく目立たないように情報を間引くことによって，画質の低下を最小限に抑えています．情報の欠損 (Loss) を大きくすれば，それだけ画質は低下しますが，ファイル･サイズは小さくなります．JPEG では，視覚的にほとんど影響を与えずに原画像の 1/10 程度にデータを圧縮できると言われています．また，画質の低下に目をつぶれば，1/100 以上に圧縮することも可能です．圧縮/符号化を行うときに，画質を優先して圧縮率を低く設定することもできますし，ファイル･サイズを優先して圧縮率を高く設定することもできます．

可逆圧縮 (予測処理型) クラスは，原画像の情報を失わないように圧縮/符号化します．復号化によって完全に原画像に戻すことができますが，圧縮率が低い欠点があります．画像の種類によっても異なりますが，JPEG の可逆圧縮ではせいぜい 1/2 程度にしか圧縮できません．可逆圧縮には JPEG 以外にもいろいろな方式があり，JPEG の可逆圧縮はあまり使われていないようです．

〈表1-3〉四つの代表的な符号化プロセス

	ベースライン・プロセス(基本DCT利用型プロセス)	拡張DCT利用型プロセス	可逆プロセス	階層プロセス
フレーム数	1	1	1	複数
符号化クラス	非可逆（DCT利用型）	非可逆（DCT利用型）	可逆（予測処理型）	拡張DCT，可逆，またはその組み合わせ
階調（各コンポーネント）	8ビット	8ビットまたは12ビット	2～16ビット	
符号化モード	シーケンシャル	シーケンシャルまたはプログレッシブ	シーケンシャル	
符号化手順	ハフマン符号化	ハフマン符号化または算術符号化	ハフマン符号化または算術符号化	
符号化テーブル	DC × 2，AC × 2	DC × 4，AC × 4	DC × 4	
符号化順序	インタリーブまたはノンインタリーブ	インタリーブまたはノンインタリーブ	インタリーブまたはノンインタリーブ	インタリーブまたはノンインタリーブ

〈図1-6〉プログレッシブ・モード



● シーケンシャル・モード，プログレッシブ・モード，可逆モード，階層モード

非可逆圧縮（DCT利用型）では，シーケンシャル・モードとプログレッシブ・モードの二つのモードが利用できます（図1-6）．

シーケンシャル・モードは，画像全体を左から右に，上から下に順次スキャンして圧縮/符号化します．復号化/再生時にも，同じ順序で端のほうから再生します．

プログレッシブ・モードは，左から右に，上から下にという順序は変わりませんが，間を飛ばしながら複数回のスキャンを行ってデータを圧縮/符号化します．復号化/再生時には，まず全体の画像を低解像度で再生し，次第に高解像度になるように再生を行います．GIFで**インタレースGIF**と呼ばれているのと同様の機能です．

可逆圧縮は，可逆モードという一つのモードになっています．スキャンはシーケンシャルだけで，プログレッシブなスキャンはできません．

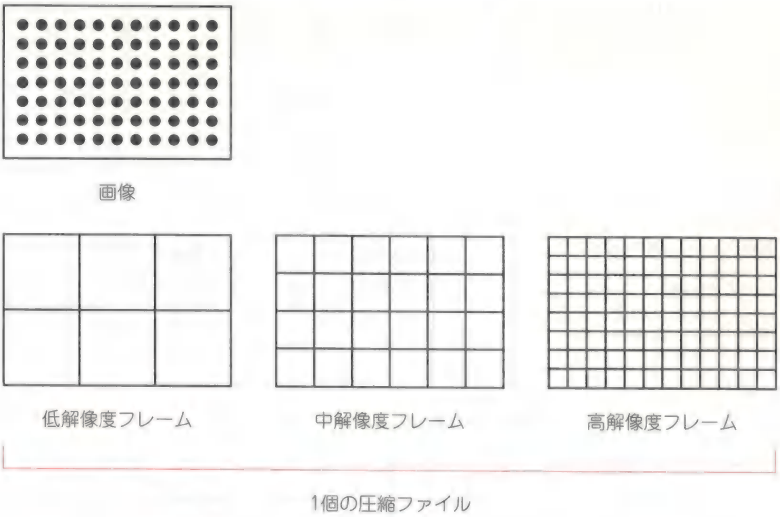
階層（ハイアラーキカル）モードは，他の三つのモードとはかなり異なります（図1-7）．これは，解像度の異なる複数の画像が必要とされる用途向けに考えられたもので，たとえば画像検索のためのサムネール画像を元の画像とまとめて圧縮/符号化したい場合に利用できます．

他のモードでは原画像を1個のフレームとして圧縮/符号化します．階層モードは，原画像を解像度が順次変化するフレームの列として圧縮/符号化します．したがって，階層モードで生成した画像ファイルは複数のフレームから構成されます．他のモードで生成したファイルはすべて単一のフレームです．各フレームの圧縮/符号化には，非可逆圧縮と可逆圧縮のどちらも使うことができます．

インタレースGIF

電話回線などの低速伝送路を通じて画像データをやり取りするとき，全部のデータを転送し終わるまでにはかなり時間がかかる．そんなとき，プログレッシブJPEGやインタレースGIFは，最初に全体の概略がわかる程度のおおざっぱなデータを送り，そこから少しずつ詳しくなっていくようにデータを送る．データを待つユーザにとっては，全体の概略だけでも早くわかるほうが，待たされる感じが少なくてすむ．

〈図 1-7〉階層モード



インタリーブ・スキャンとノンインタリーブ・スキャン

すでに述べたように，カラー画像のように複数コンポーネントから構成される画像データを圧縮/符号化するとき，インタリーブとノンインタリーブのいずれかの符号化順序を選ぶことができます．

ノンインタリーブ・スキャンでは，1 個のスキャンに 1 個のコンポーネントのデータだけを含みます．画像ファイルはコンポーネント数だけのスキャンによって構成されます．インタリーブ・スキャンでは，1 個のスキャンにすべてのコンポーネントの成分が含まれます．画像ファイルは単一のスキャンで構成されます．

ハフマン符号化手順と算術符号化手順

非可逆または可逆方式で圧縮されたデータは最終的に符号化され，それによってデータはさらに圧縮されます．**ハフマン符号**を使う符号化（Huffman coding）と，算術符号を使う符号化（Arithmetic coding）が規定されています．

ハフマン符号
データ圧縮技術の中でもっとも有名なものの一つ．もともとはデータ圧縮というより情報理論の研究から生まれたもので，いろいろなところで使われている．情報のもつエントロピーを理論的限界まで小さくできることと，ツリー・モデルを使って単純なアルゴリズムで符号化できることが特徴．

ハフマン符号化は，圧縮されたデータ・パターンに 1 ～ 16 ビットの可変長符号を割り当てて符号化するものです．出現頻度が高いパターンに短い符号を割り当てることにより，データを圧縮できます．なお，JPEG で使われているハフマン符号化は，基本的なハフマン符号化に予測処理（差分処理）や RLE（ラン・レングス・エンコード）などの手法を組み合わせた拡張ハフマン方式です．

ハフマン符号化では，パターンと符号の対応付けは可変です．復号化するときには，符号化するときに使った対応表（ハフマン・テーブル）が必要です．符号化したデータに，このハフマン・テーブルを付加します．1 個の画像の中でも，必要に応じて複数のハフマン・テーブルを使い分けることが可能です．扱えるテーブル数の上限は，ベースライン・プロセスで 4 個（2 個の DC テーブルと 2 個の AC テーブル），その他のプロセスでは 4 個または 8 個です．

算術符号化は，再帰的な確率演算によってパターンに符号を割り当てる方式です．ハフマン・テーブルのかわりに，条件付けテーブルと呼ばれるテーブルを使用して符号化/復号化を行います．圧縮効率は良いと言われていますが，**特許のライセンス**が必要なため，ハフマン符号化のほうが一般的に使われているようです．

特許のライセンス
JPEG 規格書の最後に関係する特許のリストがあるが，大部分は算術符号化に関するもの．IBM，AT&T，三菱電機などが特許をもっているようである．

四つの代表的な符号化プロセス

ベースライン・プロセス（基本DCT利用型プロセス），拡張DCT利用型プロセス，可逆プロセス，階層プロセスに分類されます．

ベースライン・プロセスは，すべてのDCT利用型のデコーダに必須とされる処理です．また，実際にもっとも広く使われています．DCTを利用するデコーダは，かならずベースライン・プロセスを実装しなければなりません．

ベースライン・プロセス以外のDCT利用型の処理は，すべて拡張DCT利用型プロセスと呼ばれます．プログレッシブ・モードや算術符号化を利用したい場合，4個以上のハフマン・テーブルを利用したい場合は，拡張DCT利用型プロセスになります．

なお，可逆プロセス専用のデコーダには，ベースライン・プロセスを実装する必要はありません．

符号化のためのベースライン・プロセス

原画像データの符号化プロセスと画像再生プロセス

JPEGでもっとも一般的に使われているベースライン・プロセス（非可逆圧縮クラス，シーケンシャル・モード，ハフマン符号化手順）の概要を示します（図1-8）．

画像データを圧縮/符号化するには，レベルシフト→順方向DCT→量子化とジグザグ・シーケンス化→ハフマン符号化という手順をとります．

符号化データを復号化/再生する手順はその逆で，ハフマン復号化→8×8配列化と逆量子化→逆方向DCT→逆レベルシフトという手順をとります．

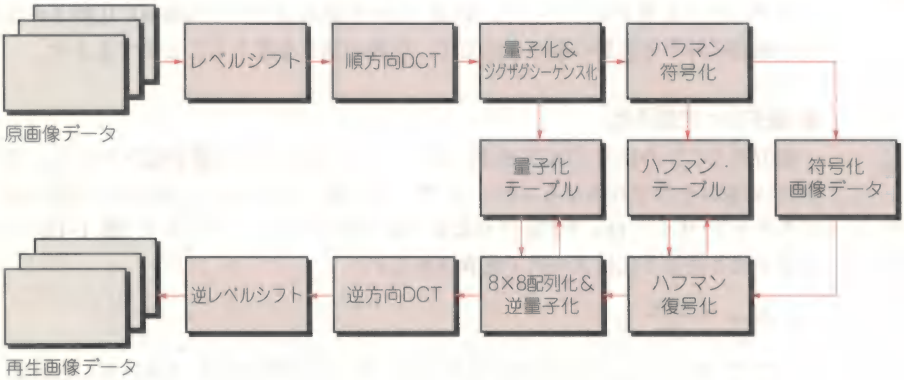
いずれの場合も，1個のスキャンを一連のデータとして符号化，復号化します．1個のフレームが複数のスキャンから構成される場合は，スキャンの数だけ符号化データのブロックができます．

量子化と逆量子化は同じ量子化テーブルを参照します．エンコーダは適当な量子化テーブルを作成して量子化を実行し，画像データに量子化テーブルを付加します．デコーダは画像データに付加された**量子化テーブル**を参照して逆量子化を実行します．

それと同様に，ハフマン符号化とハフマン復号化も同じハフマン・テーブルを参照します．エンコーダは適当なハフマン・テーブルを作成して符号化を実行し，画像データにハフマン・テーブルを付加します．デコーダは画像データに付加されたハフマン・テーブルを参照して復号化を実行します．

量子化テーブル
JPEGでは，符号化のときに使った量子化テーブルやハフマン・テーブルを画像データに付加して送る．一見データが増えそうだが，画像データ全体から見ればこれらのテーブルは小さなものだし，テーブルの分を含めても高い圧縮率が得られる．

〈図1-8〉JPEGベースライン・プロセス



● レベルシフトで正負のデータにする

DCT（離散コサイン変換）の準備として，0～255の範囲である8ビットの画像データから128を減算し，－128～+127の範囲にレベルシフトします．また復号化/再生のときは，－128～+127の範囲である逆方向DCTの出力に128を加算し，0～255の範囲に戻します．

● 順方向DCTと逆方向DCT

順方向DCTでは，8×8ブロック内の64個のデータ syx（$x, y = 0 \sim 7$）に対して離散コサイン変換を行い，64個のデータ Svu（$u, v = 0 \sim 7$）を生成します．この Svu をDCT係数と呼びます．また逆方向DCTでは，64個のDCT係数 Svu（$u, v = 0 \sim 7$）を，8×8ブロック内の64個のデータ Syx（$x, y = 0 \sim 7$）に変換します（**図1-9**）．

変換式は，

$$Svu = \frac{1}{4} Cu \cdot Cv \cdot \sum \sum syx \cdot \cos \left\{ \frac{(2x+1)u\pi}{16} \right\} \cdot \cos \left\{ \frac{(2y+1)v\pi}{16} \right\}$$

$$Syx = \frac{1}{4} \sum \sum Cu \cdot Cv \cdot Svu \cdot \cos \left\{ \frac{(2x+1)u\pi}{16} \right\} \cdot \cos \left\{ \frac{(2y+1)v\pi}{16} \right\}$$

ただし，$u = 0,\ v = 0$ ならば $Cu,\ Cv = 1/\sqrt{2}$，それ以外のとき $Cu,\ Cv = 1$ と表されます．

DCT係数 Svu も8×8の2次元配列（正方形のテーブル）で表されますが，その各項は元の8×8ブロックの特定のピクセルに対応するものではありません．DCT係数の各項は，すべて元の8×8ブロック全体の情報を含んでいます．

とくに $u = v = 0$ のときのDCT係数 $S00$ は，二つのcosの項がともに $\cos 0 = 1$ であり，角度の項がなくなります．すなわち，

$$S00 = \frac{1}{8} \sum \sum syx$$

です．これは，8×8ブロック内の画像データの平均値（を8倍したもの）に相当します．これを**DC係数**と呼びます．元の画像データが8ビットなのに対して，DC係数はその8倍ですから，11ビット相当の精度をもちます．

それ以外のDCT係数はすべて角度の項をもちます．これらを**AC係数**と呼びます．DCTの角度（角周波数）の項は画像上では周波数に相当しますから，AC係数は8×8ブロック内での画像データの変化を周波数成分ごとに取り出したものに相当します．u, v が大きいほど，すなわち8×8配列の右下に近づくほど，周波数は高くなります．したがって，画像の細かい変化を表す情報になります．

つぎに述べる量子化によって，64個のDCT係数はそれぞれ個別に圧縮されます．画質に対する影響の強さに応じて，圧縮の強さを変えることができます．

DC係数

このDCは直流の意味．画像データの直流成分を示すもの．DCT係数と似ていてまぎらわしい．

AC係数

このACは交流の意味．画像データの中のさまざまな周波数成分を表すもの．したがって，DC係数は1個だが，AC係数はさまざまな周波数に対応してたくさんある．

量子化

連続的なアナログ量を，不連続なディジタル量に置き換える．すなわち，四捨五入とか丸め，切り捨てということ．画像データはすでにディジタル値になっているが，さらに細部を切り捨ててデータをコンパクトにしようということ．

● 量子化と逆量子化

順方向DCTの出力（DCT係数）は，データ圧縮のため**量子化**されます．各DCT係数はそれぞれ異なるステップ・サイズで量子化できます．各DCT係数 Svu にステップ・サイズ Qvu を対応させた表を量子化テーブルと呼びます（**図1-10**）．

量子化と逆量子化は次の式で定義されます．

$$Sqvu = \frac{Svu}{Qvu}$$

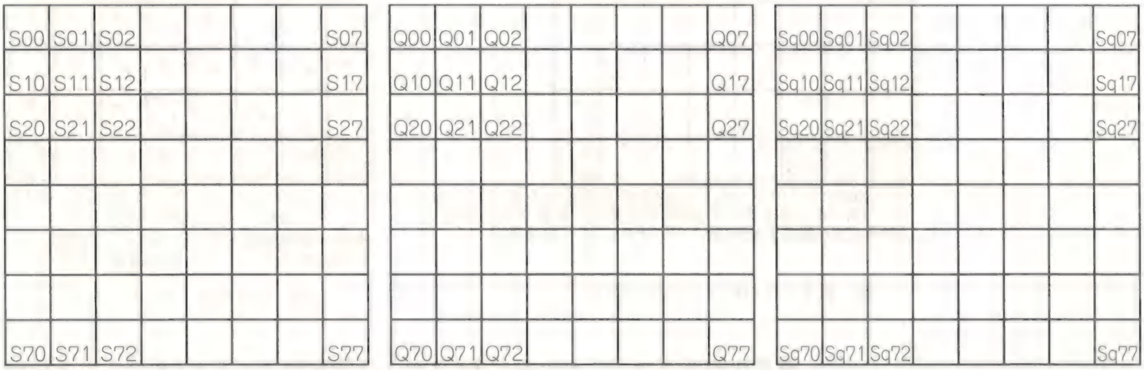
ここで，$[Svu/Qvu]$ は Svu/Qvu を超えない最大の整数を示します．Svu/Qvu の

〈図1-9〉DCT（離散コサイン変換）

8×8ブロックのサンプル値sxy

変換値Suv=(1/4)$CuCv$Σ Σsyx{cos(2x+1)uπ/16} {cos(2y+1)vπ/16}

〈図1-10〉量子化テーブル（量子化 Svu/Qvu → $Squv$，逆量子化 Svu ← $Squv$ × Qvu）



DCT係数(8×8)　量子化テーブル(8×8)　量子化されたDCT係数(8×8)

小数点以下の端数は整数に丸められ，その分の情報が失われます．ステップ・サイズ Qvu を大きくするほど Svu/Qvu は小さな値となり，多くの情報が小数点以下になるため，演算[Svu/Qvu]によって多くの情報が切り捨てられます．64個のDCT係数のうち，画質に対する影響の大きいものはステップ・サイズを小さくしてより多くの情報を保存し，画質に対する影響の小さいものはステップ・サイズを大きくして圧縮率を高めることができます．

復号化するときには，符号化のときに使った量子化テーブルを使用して逆量子化を行います．量子化テーブルは符号化データに付加されます．

付属書Kには，YCbCr向けの量子化テーブルの例が掲載されています（**表1-4**）．

● ジグザグ・シーケンス化と8×8配列化

量子化されたDCT係数は8×8の2次元配列 $Sqvu$ （u, v=0～7）ですが，符号化の準備としてこれを64個の1次元配列 ZZi （i =0～63）にします（**図1-11**）．DC係数（Sq00）から出発し，8×8配列をジグザグにたどりながら1次元配列化します．ジグザグにたどるのは，全体として8×8配列の左上（直流～低周波成分）から右下（高周波成分）に向かって1次元のデータが並ぶようにするためです．

ジグザグ・シーケンス化

全体として8×8配列の左上から右下に向かって各係数を1列に並べればいいので，かならずしもジグザグでなければならないわけではない．たとえば，AC01とAC10は対等であり，どちらを先に並べてもいいのだが，ジグザグ・シーケンスという言葉がおもしろいのでこの順番に決められたのだろう．

〈表 1-4〉 量子化テーブルの例

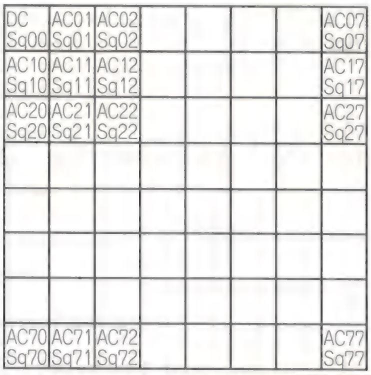
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(a) 輝度（Y）テーブル

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

(b) 色差（Cb, Cr）テーブル

〈図 1-11〉 ジグザグ・シーケンス化



Sq00は8×8ブロック内での画像の直流成分を表す（DC係数）．
それ以外の63個はすべて画像の交流成分を表す（AC係数）．
これらの64個を1次元のシーケンスに並べ換える．

(a) 量子化されたDCT係数（8×8）

ZZi ＝(DC,AC00,AC01,AC10,AC20,AC11,
AC02,…AC67,AC76,AC77)
直流～低周波成分がシーケンスの前寄りに，
高周波成分がシーケンスの後寄りに並ぶ．

(b) ジグザグ・シーケンス化

復号化のときは，これと逆に 64 個の 1 次元配列 *ZZi* を 8×8 の 2 次元配列 *Sqvu*（*u, v* =0 ～ 7）にします．

また，DC 係数についてはここで差分化も行います．

DC 係数は 8×8 ブロック全体を代表する値であり，量子化のステップ・サイズを大きくすると画質が低下します．そのため，DC 係数を量子化してもあまり強く圧縮できません．

一方，画像の端の部分を除けば，隣接した 8 × 8 ブロックの DC 係数は比較的近い値であることが期待できます．そこで，直前に符号化した 8 × 8 ブロックの DC 係数（すなわち，隣接するブロックの DC 係数）を予測値として，それと今回符号化する 8 × 8 ブロックの DC 係数との差分値をとることによって情報量を減らすことができます．この手法は，可逆圧縮で使われている予測処理と同様のものです．

スキャンの開始点（画像の左上）では，予測値の初期値は 0 にリセットされます．それ以後は直前の DC 係数を予測値として差分を行います．ただし，それ以後でも必要に応じて予測値を 0 にリセットすることができます．これを**リスタート**と呼びます．

たとえば，画像を左から右に 1 ライン符号化すると，次は 1 段下がって左から右に 1 ライン符号化します．このとき，前のラインの右端の DC 係数が次のラインの左端の DC 係数の予測値となりますが，この二つには相関が期待できません．1 ラインの符号化が終わるごとに（あるいは数ラインごとに）予測値をリセ

リスタート

リスタートの地点では，予測値を 0 にして減算を行うので，それは差分しないもともとの値と同じである．リスタートしてもしなくても，データ圧縮の効果は変わらない．ただ，差分予測だけを続けていくと，どこかでノイズやデータ化けの影響を受けると，それがずっと後まで残ってしまうことになる．リスタートには，そういう影響を打ち切って仕切り直すという効果もある．

〈図 1-12〉DC 係数のハフマン符号

可変長ワード（ハフマン符号）	可変長バリュー
1～16 ビット	0～11 ビット

可変長ワードと可変長バリューの組み合わせで構成される.
可変長ワード部のほうにハフマン符号を用いている.

(a)　符号の構成

カテゴリ値		可変長ワード		量子化されたDC係数の差分値		可変長バリュー	
SSSS	16進	値(*)	ビット数	Diff	個数	値	ビット数
0000	0x0	HUF0	1～16ビット	0	1	なし	0
0001	0x1	HUF1	1～16ビット	−1, 1	2	0, 1	1
0010	0x2	HUF2	1～16ビット	−3, −2, 2, 3	4	00, 01, 10, 11	2
0011	0x3	HUF3	1～16ビット	−7, …, −4, 4,…, 7	8	000,…, 011,100,…, 111	3
0100	0x4	HUF4	1～16ビット	−15, …, −8, 8, …, 15	16	0000,…, 0111,1000,…, 1111	4
0101	0x5	HUF5	1～16ビット	−31,…, −16, 16,…, 31	32	00000,…, 01111,10000,…, 11111	5
0110	0x6	HUF6	1～16ビット	−63,…, −32, 32,…, 63	64	000000,…, 011111,100000,…, 111111	6
0111	0x7	HUF7	1～16ビット	−127,…, −64, 64,…, 127	128	0000000,…, 0111111,1000000,…, 1111111	7
1000	0x8	HUF8	1～16ビット	−255,…, −128,128,…, 255	256	00000000,…, 01111111,10000000,…, 11111111	8
1001	0x9	HUF9	1～16ビット	−511,…, −256,256,…, 511	512	000000000,…, 011111111,100000000,…, 111111111	9
1010	0xA	HUF10	1～16ビット	−1023,…, −512,512,…, 1023	1024	0000000000,…, 0111111111,1000000000,…, 1111111111	10
1011	0xB	HUF11	1～16ビット	−2047,…, −1024,1024,…, 2047	2048	00000000000,…, 01111111111,10000000000,…, 11111111111	11

(*) HUF0～HUF11は，1～16ビットのハフマン符号から選んだ任意の12個．出現頻度の高いカテゴリに短いワードを割り当てれば圧縮効率が向上する.
量子化されたDC係数の差分をとることによって，−2047～2047の範囲の差分値が得られる.
差分値は0に近いものから順に12のカテゴリに分けられ，各カテゴリに可変長ワードが割り当てられる.
各カテゴリ内で複数の差分値を区別するために，可変長バリューが割り当てられる.

(b)　カテゴリ値と差分値

ットしてリスタートすることができます.

復号化のときは，差分値からDC係数を復元します.

● DC係数のハフマン符号化とハフマン復号化

量子化され，1次元に並べられたDCT係数の列は，ハフマン・テーブルを参照して符号化されます．このとき，DC係数とAC係数は異なるハフマン・テーブルを使います．64個のジグザグ・シーケンスの最初の1個（ZZ0）はDCハフマン・テーブルを使って，続く63個（ZZ1～ZZ63）はACハフマン・テーブルを使って処理されます.

また，画像データが複数のスキャンで構成されている場合，スキャンごとにハフマン・テーブルを切り替えられます．DC係数，AC係数とも，ベースライン・プロセスではそれぞれ2個まで，拡張DCT利用型プロセスや可逆プロセスではそれぞれ4個までのハフマン・テーブルを使い分けられます.

DC係数のハフマン符号は，1～16ビットの**可変長ワード**とそれに続く0～11ビットの**可変長バリュー**で構成されます（**図1-12**）．まず，DC係数の差分値Diffの絶対値が12のカテゴリに分類されます．それぞれのカテゴリ値SSSS（4ビット）に対して可変長ワードが割り当てられ，各カテゴリに含まれる差分値Diffを区別するために可変長バリューが割り当てられます．これによって，差分値Diffと1対1に対応する可変長符号が得られます.

ハフマン・テーブルはカテゴリ値に可変長ワードを対応させるものです．テーブルの前半は，1ビットから16ビットまで，各ビット長のワードが何個使われるかを示します．テーブルの後半で，使われるワードに順次カテゴリ値を割り当てます（**図1-13**）．出現頻度の高いカテゴリ値に短いワードを割り当てれば，圧縮率を高くできます.

nビットの2進数は2^n個ありますが，その中から，より短いワードとビッ

可変長ワード
本来のハフマン符号はこの可変長ワードのほうである．テーブルを参照して符号化，復号化を行うことが必要.

可変長バリュー
こちらはテーブルを使わずに，機械的な計算で求められる部分.

ト・パターンが重複するものと，すべてのビットが1であるものが除外されます．すべてのビットが1のものは，より長いワードのためのプリフィックスとして予約されます．

たとえば，1ビットの2進数は0と1があります．1は予約されているため，1ビットのワードとして使えるのは0だけです．

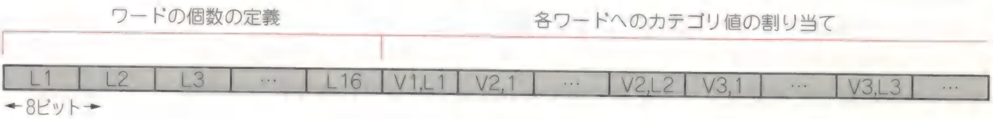
〈図1-13〉DC係数のハフマン・テーブル

1ビット		2ビット		3ビット		4ビット		トータル
個数	ワード	個数	ワード	個数	ワード	個数	ワード	個数
1	0	1	10	1	110	1	1110	4
1	0	1	10			3	1100,1101,1110	5
1	0			3	100,101, 110	1	1110	5
1	0			2	100,101	3	1100,1101,1110	6
1	0			1	100	5	1010,1011,1100,1101,1110	7
1	0					7	1000,1001,1010,1011,1100,1101,1110	8
		3	00, 01, 10	1	110	1	1110	5
		3	00, 01, 10			3	1100,1101,1110	6
		2	00, 01	3	100,101, 110	1	1110	6
		2	00, 01	2	100,101	3	1100,1101,1110	7
		2	00, 01	1	100	5	1010,1011,1100,1101,1110	8
		2	00, 01			7	1000,1001,1010,1011,1100,1101,1110	9
		1	00	5	010,011, 100,101, 110	1	1110	7
		1	00	4	010,011, 100,101	3	1100,1101,1110	8
		1	00	3	010,011, 100	5	1010,1011,1100,1101,1110	9
		1	00	2	010,011	7	1000,1001,1010,1011,1100,1101,1110	10
		1	00	1	010	9	0110,0111,1000,1001,1010,1011,1100,1101,1110	11
		1	00			11	0100,0101,0110,0111,1000,1001,1010,1011,1100,1101,1110	12
				7	000,001, 010,011, 100,101, 110	1	1110	8
				6	000,001, 010,011, 100,101	3	1100,1101,1110	9
				5	000,001, 010,011, 100	5	1010,1011,1100,1101,1110	10
				4	000,001, 010,011	7	1000,1001,1010,1011,1100,1101,1110	11
				3	000,001, 010	9	0110,0111,1000,1001,1010,1011,1100,1101,1110	12
				2	000,001	11	0100,0101,0110,0111,1000,1001,1010,1011,1100,1101,1110	13
				1	000	13	0010,0011,0100,0101,0110,0111,1000,1001,1010,1011,1100,1101,1110	14
						15	0000,0001,0010,0011,0100,0101,0110,0111,1000,1001,1010,1011,1100,1101,1110	15

4ビットの場合のハフマン符号の選び方の例を示す．
すべてのビットが1のものは，より長いワードのためのプリフィックスとして予約される．
一般に，Nビットのワードを1個選ぶと，$N+1$ビットで選べるワードは2個減る．
どのようにワードを選んでいるかは，各ビットのワードの個数だけで決まる．

- 1ビットで選べるのは0だけ．
  1ビットのワードを1個選ぶと，2～4ビットで0で始まるものは選べない．
- 2ビットでは00，01，10が選べる．
  1ビットで1個選ぶと，2ビットで選べるのは10だけ．
  1ビットが0個だと，2ビットで00，01，10を選べる．
  1ビットで1個，2ビットで1個選ぶと，3～4ビットで10で始まるものは選べない．
  2ビットで3個選ぶと，3～4ビットで10で始まるものは選べない．
- 3ビットでは000，001，010，011，100，101，110が選べる．
  1ビットで1個，2ビットで1個選ぶと，3ビットで選べるのは110だけ．
  2ビットで3個選ぶと，3ビットで選べるのは110だけ．
  2ビットで2個選ぶと，3ビットで100，101，110が選べる．
  2ビットで1個選ぶと，3ビットで010，011，100，101，110が選べる．

(a) 4ビットのハフマン符号の選び方の例



(b) ハフマン・テーブルの構成

0x00	0x01	0x05	0x01	0x01	0x01	0x01	0x01	0x01	0x00	0x00	0x00	0x00
L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13

(c) ハフマン・テーブルの数値列の例（このハフマン・テーブルによって，**表1-5**(a)

〈図 1-14〉可変長バリューの付加

カテゴリ値	Diff（10 進）	Diff（2 進）	Diff − 1（負数のみ）	可変長バリュー（2 進）
1	− 1	111111111111	111111111110	0
	1	000000000001		1
2	− 3	111111111101	111111111100	00
	− 2	111111111110	111111111101	01
	2	000000000010		10
	3	000000000011		11
3	− 7	111111111001	111111111000	000
	− 6	111111111010	111111111001	001
	− 5	111111111011	111111111010	010
	− 4	111111111100	111111111011	011
	4	000000000100		100
	5	000000000101		101
	6	000000000110		110
	7	000000000111		111
4	− 15	111111110001	111111110000	0000
	− 14	111111110010	111111110001	0001
	− 13	111111110011	111111110010	0010
	− 12	111111110100	111111110011	0011
	− 11	111111110101	111111110100	0100
	− 10	111111110110	111111110101	0101
	− 9	111111110111	111111110110	0110
	− 8	111111111000	111111110111	0111
	8	000000001000		1000
	9	000000001000		1001
	10	000000001000		1010
	11	000000001000		1011
	12	000000001000		1100
	13	000000001000		1101
	14	000000001000		1110
	15	000000001000		1111

(a) 可変長バリューの計算

00	Diff=0
010	0	Diff= − 1
010	1	Diff=1
011	00	Diff= − 3
011	01	Diff= − 2
011	10	Diff=2
011	11	Diff=3
100	000	Diff= − 7
100	001	Diff= − 6
100	010	Diff= − 5
100	011	Diff= − 4
100	100	Diff=4
100	101	Diff=5
100	110	Diff=6
100	111	Diff=7

↑可変長ワード　↑可変長バリュー

カテゴリ 0 に 00，カテゴリ 1 に 010，カテゴリ 2 に 011，カテゴリ 3 に 100 の可変長ワードを割り当てた例

(b) 可変長バリューの付加

| V16,1 | … | V16,L16 |

Liは，iビットのワードの個数
L1=0～1
L2=0～3
L3=0～7
…
L16=0～65535
ただし，L1+L2+…=12

Vi,jは，iビットのワードのj番目のものに割り当てられるカテゴリ値
Vij =0～11

iビットのワードが0個(Li=0)なら，そのiに対するVijは省略される

0x00	0x00	0x00	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B
L14	L15	L16	V2,1	V3,1	V3,2	V3,3	V3,4	V3,5	V4,1	V5,1	V6,1	V7,1	V8,1	V9,1

輝度 DC テーブルの割り当てが定義される)

2ビットの2進数は00, 01, 10, 11がありますが，11は予約されています．また，0を1ビットのワードとして使うなら，1ビット目がそれと重複する00, 01が除外され，2ビットのワードとして使えるのは10だけになります．もし，0を1ビットのワードとして使わなければ，2ビットのワードとして00, 01, 10の3

〈表1-5〉DCハフマン・テーブルの例

SSSS	可変長ワード	ワード長
0x0	00	2
0x1	010	3
0x2	011	3
0x3	100	3
0x4	101	3
0x5	110	3
0x6	1110	4
0x7	11110	5
0x8	111110	6
0x9	1111110	7
0xA	11111110	8
0xB	111111110	9

(a) 輝度DCテーブルの符号割り当て例

SSSS	可変長ワード	ワード長
0x0	00	2
0x1	01	2
0x2	10	2
0x3	110	3
0x4	1110	4
0x5	11110	5
0x6	111110	6
0x7	1111110	7
0x8	11111110	8
0x9	111111110	9
0xA	1111111110	10
0xB	11111111110	11

(b) 色差DCテーブルの符号割り当て例

〈図1-15〉AC係数のハフマン符号

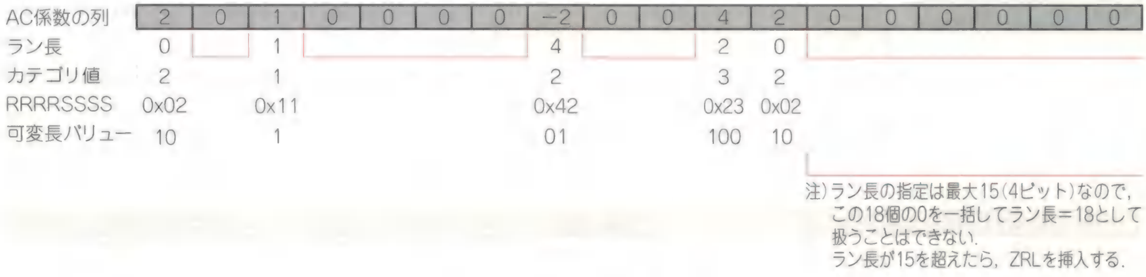
可変長ワード（ハフマン符号）	可変長バリュー
1～16ビット	0～11ビット

可変長ワードと可変長バリューの組み合わせで構成される．
可変長ワード部のほうにハフマン符号を用いている．

(a) 符号の構成

ラン長	カテゴリ値	16進	可変長ワード		量子化されたAC係数		可変長バリュー	
RRRR	SSSS	RRRRSSSS	値(*)	ビット数	AC	個数	値	ビット数
0000	0000	0x00	HUF0	1～16ビット			なし	0
1111	0000	0xF0	HUF1	1～16ビット			なし	0
0000～1111	0001	0x01～0xF1	HUF2～HUF17	1～16ビット	−1, 1	2	0, 1	1
0000～1111	0010	0x02～0xF2	HUF18～HUF33	1～16ビット	−3, −2, 2,3	4	00, 01, 10, 11	2
0000～1111	0011	0x03～0xF3	HUF34～HUF49	1～16ビット	−7,…, −4, 4,…, 7	8	000,…, 011,100,…, 111	3
0000～1111	0100	0x04～0xF4	HUF50～HUF65	1～16ビット	−15,…, −8, 8,…, 15	16	0000,…, 0111,1000,…, 1111	4
0000～1111	0101	0x05～0xF5	HUF66～HUF81	1～16ビット	−31,…, −16, 16,…, 31	32	00000,…, 01111,10000,…, 11111	5
0000～1111	0110	0x06～0xF6	HUF82～HUF97	1～16ビット	−63,…, −32, 32,…, 63	64	000000,…, 011111,100000,…, 111111	6
0000～1111	0111	0x07～0xF7	HUF98～HUF113	1～16ビット	−127,…, −64, 64,…, 127	128	0000000,…, 0111111,1000000,…, 1111111	7
0000～1111	1000	0x08～0xF8	HUF114～HUF129	1～16ビット	−255,…, −128,128,…, 255	256	00000000,…, 01111111,10000000,…, 11111111	8
0000～1111	1001	0x09～0xF9	HUF130～HUF145	1～16ビット	−511,…, −256,256,…, 511	512	000000000,…, 011111111,100000000,…, 111111111	9
0000～1111	1010	0x0A～0xFA	HUF146～HUF161	1～16ビット	−1023,…, −512,512,…, 1023	1024	0000000000,…, 0111111111,1000000000,…, 1111111111	10

(b) カテゴリ値，ラン長とAC係数



(c) ラン長と0でないAC係数

個が使えます．

さらに，各カテゴリ内で差分値 Diff を区別するための可変長バリューが付加されます．カテゴリ n のワードには，差分値 Diff が正なら Diff の下位 n ビット，Diff が負なら Diff−1 の下位 n ビットが付加されます．これによって，各差分値と 1 対 1 に対応する**可変長符号**が得られます（**図 1-14**）．

付属書 K には，YCbCr 向けの DC テーブルの例が二つ掲載されています．**表 1-5**(a)は輝度情報，**表 1-5**(b)は色差情報に適した DC テーブルの例です．

たとえば，**表 1-5**(a)の DC テーブルを使えば，差分値 Diff＝0 は 00，Diff＝＋1 は 0101，Diff＝−1 は 0100，Diff＝＋2 は 01110，Diff＝−2 は 01101，Diff＝＋3 は 01111，Diff＝−3 は 01100 というように符号化されます．

また，**表 1-5**(b)の DC テーブルを使えば，Diff＝0 は 00，Diff＝＋1 は 011，Diff＝−1 は 010，Diff＝＋2 は 1010，Diff＝−2 は 1001，Diff＝＋3 は 1011，Diff＝−3 は 1000 というように符号化されます．

デコーダは DC ハフマン・テーブルを参照してワードをカテゴリに変換し，ワードに続く可変長バリューを使って差分値 Diff を復号化します．

可変長符号

可変長ワードと可変長バリューを組み合わせることにより，単なるハフマン符号化よりも複雑な手順で作り出している．そのかわり，テーブルを小さくすることができる．

● AC 係数のハフマン符号化とハフマン復号化

AC 係数のハフマン符号化/復号化はさらに複雑に構成されています（**図 1-15**）．

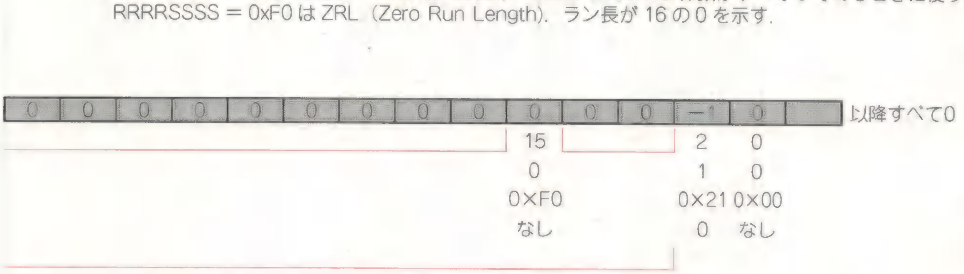
まず，AC 係数の値は 0 のものが比較的多いので，AC 係数の値が 0 なら符号化せず，0 でない AC 係数だけを 10 のカテゴリに分類して符号化します．どの AC 係数が 0 でないかを示すために，**ラン長**（Run Length）の考え方が使われます．その前に符号化した 0 でない AC 係数と，今回符号化する 0 でない AC 係数の間にはさまれた 0 の個数をラン長と呼びます．

AC 係数のハフマン符号は，1 ～ 16 ビットの可変長ワードとそれに続く 1 ～ 10 ビットの可変長バリューで構成されます．まず，0 でない AC 係数を絶対値で分類して 1 ～ 10 のカテゴリに分けたカテゴリ値 SSSS（4 ビット）に，その前の 0 でない AC 係数からのラン長 RRRR（4 ビット）を組み合わせて，8 ビットの

ラン長

これも，データ圧縮の手法としてはとてもよく使われるもの．本来は，2 値画像のように同じ値が長く続くデータで効果を上げられるが，AC 係数の場合は 0 が長く続くので，このラン長との組み合わせが効果をもつ．

(*) HUF0 ～ HUF161 は，1 ～ 16 ビットのハフマン符号から選んだ任意の 162 個．
量子化された AC 係数の値は，− 1023 ～ 1023 の範囲．
AC 係数 0 は符号化しない．そのかわり，AC 係数＝0 のラン長 RRRR を，次の 0 でないカテゴリ SSSS と合わせて符号化する．
AC 係数は 0 に近いものから順に 11 のカテゴリに分けられ，ラン長と各カテゴリの組み合わせに対して可変長ワードが割り当てられる．
各カテゴリ内で複数の差分値を区別するために，可変長バリューが割り当てられる．
SSSS＝0 には特別なコードが 2 個割り当てられている．
RRRRSSSS＝0x00 は EOB（End Of Block）．そこから先の AC 係数がすべて 0 であるときに使う．
RRRRSSSS＝0x00 は EOB（End Of Block）．そこから先の AC 係数がすべて 0 であるときに使う．
RRRRSSSS＝0xF0 は ZRL（Zero Run Length）．ラン長が 16 の 0 を示す．



データ RRRRSSSS を作ります．この RRRRSSSS に可変長ワードを割り当て，さらに DC 係数の場合と同様に，カテゴリに含まれる複数の AC 係数を区別するための可変長バリューを付加します．

0 でない AC 係数だけを符号化するので，SSSS が 0 になることはありません．ただし，二つの特別な場合だけ SSSS＝0 として符号化を行います．一つは，ラン長が 15 個を超える場合です．このときは，16 個目の 0 を RRRRSSSS＝0 xF0 として符号化します．

もう一つは，ある番号から先の AC 係数がすべて 0 になる場合です．このときは，最後の 0 でない AC 係数に続く 0 を RRRRSSSS＝0 x00 として符号化し，そこで符号化を打ち切ります．デコーダが 0 x00 を検出したら，そこから ZZ63 までのすべての AC 係数を 0 に復号化します．この 0 xF0 を ZRL(Zero Run Length)，0 x00 を EOB (End Of Block) と呼びます．

付属書 K には，YCbCr 向けの AC テーブルの例が二つ掲載されています．**表 1-6**(a)は輝度情報，**表 1-6**(b)は色差情報に適した AC テーブルの例です．

〈表 1-6〉 AC 係数のハフマン・テーブルの例

RRRRSSSS	可変長ワード	ワード長	RRRRSSSS	可変長ワード	ワード長	RRRRSSSS	可変長ワード	ワード長	RRRRSSSS	可変長ワード	ワード長
0x00 (EOB)	1010	4	0 x40	—	—	0 x80	—	—	0 xC0	—	—
0x01	00	2	0 x41	111011	6	0 x81	111111000	9	0 xC1	1111111010	10
0x02	01	2	0 x42	1111111000	10	0 x82	111111111000000	15	0 xC2	1111111111011001	16
0x03	100	3	0 x43	1111111110010110	16	0 x83	1111111110110110	16	0 xC3	1111111111011010	16
0x04	1011	4	0 x44	1111111110010111	16	0 x84	1111111110110111	16	0 xC4	1111111111011011	16
0x05	11010	5	0 x45	1111111110011000	16	0 x85	1111111110111000	16	0 xC5	1111111111011100	16
0x06	1111000	7	0 x46	1111111110011001	16	0 x86	1111111110111001	16	0 xC6	1111111111011101	16
0x07	11111000	8	0 x47	1111111110011010	16	0 x87	1111111110111010	16	0 xC7	1111111111011110	16
0x08	1111110110	10	0 x48	1111111110011011	16	0 x88	1111111110111011	16	0 xC8	1111111111011111	16
0x09	1111111110000010	16	0 x49	1111111110011100	16	0 x89	1111111110111100	16	0 xC9	1111111111100000	16
0x0A	1111111110000011	16	0 x4A	1111111110011101	16	0 x8A	1111111110111101	16	0 xCA	1111111111100001	16
0x10	—	—	0 x50	—	—	0 x90	—	—	0 xD0	—	—
0x11	1100	4	0 x51	1111010	7	0 x91	111111001	9	0 xD1	11111111000	11
0x12	11011	5	0 x52	11111110111	11	0 x92	1111111110111110	16	0 xD2	1111111111100010	16
0x13	1111001	7	0 x53	1111111110011110	16	0 x93	1111111110111111	16	0 xD3	1111111111100011	16
0x14	111110110	9	0 x54	1111111110011111	16	0 x94	1111111111000000	16	0 xD4	1111111111100100	16
0x15	11111110110	11	0 x55	1111111110100000	16	0 x95	1111111111000001	16	0 xD5	1111111111100101	16
0x16	1111111110000100	16	0 x56	1111111110100001	16	0 x96	1111111111000010	16	0 xD6	1111111111100110	16
0x17	1111111110000101	16	0 x57	1111111110100010	16	0 x97	1111111111000011	16	0 xD7	1111111111100111	16
0x18	1111111110000110	16	0 x58	1111111110100011	16	0 x98	1111111111000100	16	0 xD8	1111111111101000	16
0x19	1111111110000111	16	0 x59	1111111110100100	16	0 x99	1111111111000101	16	0 xD9	1111111111101001	16
0x1A	1111111110001000	16	0 x5A	1111111110100101	16	0 x9A	1111111111000110	16	0 xDA	1111111111101010	16
0x20	—	—	0 x60	—	—	0 xA0	—	—	0 xE0	—	—
0x21	11100	5	0 x61	1111011	7	0 xA1	111111010	9	0 xE1	1111111111101011	16
0x22	11111001	8	0 x62	111111110110	12	0 xA2	1111111111000111	16	0 xE2	1111111111101100	16
0x23	1111110111	10	0 x63	1111111110100110	16	0 xA3	1111111111001000	16	0 xE3	1111111111101101	16
0x24	111111110100	12	0 x64	1111111110100111	16	0 xA4	1111111111001001	12	0 xE4	1111111111101110	16
0x25	1111111110001001	16	0 x65	1111111110101000	16	0 xA5	1111111111001010	16	0 xE5	1111111111101111	16
0x26	1111111110001010	16	0 x66	1111111110101001	16	0 xA6	1111111111001011	16	0 xE6	1111111111110000	16
0x27	1111111110001011	16	0 x67	1111111110101010	16	0 xA7	1111111111001100	16	0 xE7	1111111111110001	16
0x28	1111111110001100	16	0 x68	1111111110101011	16	0 xA8	1111111111001101	16	0 xE8	1111111111110010	16
0x29	1111111110001101	16	0 x69	1111111110101100	16	0 xA9	1111111111001110	16	0 xE9	1111111111110011	16
0x2A	1111111110001110	16	0 x6A	1111111110101101	16	0 xAA	1111111111001111	16	0 xEA	1111111111110100	16
0x30	—	—	0 x70	—	—	0 xB0	—	—	0 xF0 (ZRL)	11111111001	11
0x31	111010	6	0 x71	11111010	8	0 xB1	111111001	10	0 xF1	1111111111110101	16
0x32	111110111	9	0 x72	111111110111	12	0 xB2	1111111111010000	16	0 xF2	1111111111110110	16
0x33	111111110101	12	0 x73	1111111110101110	16	0 xB3	1111111111010001	16	0 xF3	1111111111110111	16
0x34	1111111110001111	16	0 x74	1111111110101111	16	0 xB4	1111111111010010	16	0 xF4	1111111111111000	16
0x35	1111111110010000	16	0 x75	1111111110110000	16	0 xB5	1111111111010011	16	0 xF5	1111111111111001	16
0x36	1111111110010001	16	0 x76	1111111110110001	16	0 xB6	1111111111010100	16	0 xF6	1111111111111010	16
0x37	1111111110010010	16	0 x77	1111111110110010	16	0 xB7	1111111111010101	16	0 xF7	1111111111111011	16
0x38	1111111110010011	16	0 x78	1111111110110011	16	0 xB8	1111111111010110	16	0 xF8	1111111111111100	16
0x39	1111111110010100	16	0 x79	1111111110110100	16	0 xB9	1111111111010111	16	0 xF9	1111111111111101	16
0x3A	1111111110010101	16	0 x7A	1111111110110101	16	0 xBA	1111111111011000	16	0 xFA	1111111111111110	16

(a) 輝度 AC テーブルの符号割り当て例

JPEG ファイルの基本フォーマット

● JPEG 画像ファイルのフォーマット

JPEG のエンコーダは，圧縮/符号化した画像データにさまざまな付加情報を付け加えて画像ファイルを構成します．デコーダは，画像ファイルから付加情報を分離し，その情報を使って符号化データを復号化/再生します．

画像ファイルの基本的なフォーマットは JPEG 規格で規定されています．これを交換フォーマットと呼びます．画像ファイルは符号化データとヘッダからなります．ヘッダには，復号化/再生に必要なさまざまな付加情報が含まれます．さらに，量子化テーブルやハフマン・テーブルなどのテーブルも付加されます．

テーブル情報なしで画像データだけを示す省略フォーマットや，画像データなしでテーブル情報だけを示す省略フォーマットなどの規定もあります．

交換フォーマット

異なるシステム間でデータを交換するために取り決めたフォーマット，というような意味．生の画像データだけを異なるシステムに転送しても，受け取った側ではそれをどのように解釈し，表示してよいかわからない．そのため，データの解釈や表示に必要なパラメータをデータに付加して送る必要がある．

RRRRSSSS	可変長ワード	ワード長	RRRRSSSS	可変長ワード	ワード長	RRRRSSSS	可変長ワード	ワード長	RRRRSSSS	可変長ワード	ワード長
0x00 (EOB)	00	2	0x40	—	—	0x80	—	—	0xC0	—	—
0x01	01	2	0x41	111010	6	0x81	11111001	8	0xC1	111111010	9
0x02	100	3	0x42	111110110	9	0x82	1111111110110111	16	0xC2	1111111111011011	16
0x03	1010	4	0x43	1111111110010111	16	0x83	1111111110111000	16	0xC3	1111111111011100	16
0x04	11000	5	0x44	1111111110011000	16	0x84	1111111110111001	16	0xC4	1111111111011101	16
0x05	11001	5	0x45	1111111110011001	16	0x85	1111111110111010	16	0xC5	1111111111011110	16
0x06	111000	6	0x46	1111111110011010	16	0x86	1111111110111011	16	0xC6	1111111111011111	16
0x07	1111000	7	0x47	1111111110011011	16	0x87	1111111110111100	16	0xC7	1111111111100000	16
0x08	111110100	9	0x48	1111111110011100	16	0x88	1111111110111101	16	0xC8	1111111111100001	16
0x09	1111110110	10	0x49	1111111110011101	16	0x89	1111111110111110	16	0xC9	1111111111100010	16
0x0A	111111110100	12	0x4A	1111111110011110	16	0x8A	1111111110111111	16	0xCA	1111111111100011	16
0x10	—	—	0x50	—	—	0x90	—	—	0xD0	—	—
0x11	1011	4	0x51	111011	6	0x91	111110111	9	0xD1	11111111001	11
0x12	111001	6	0x52	1111111001	10	0x92	1111111111000000	16	0xD2	1111111111100100	16
0x13	11110110	8	0x53	1111111110011111	16	0x93	1111111111000001	16	0xD3	1111111111100101	16
0x14	111110101	9	0x54	1111111110100000	16	0x94	1111111111000010	16	0xD4	1111111111100110	16
0x15	11111110110	11	0x55	1111111110100001	16	0x95	1111111111000011	16	0xD5	1111111111100111	16
0x16	111111110101	12	0x56	1111111110100010	16	0x96	1111111111000100	16	0xD6	1111111111101000	16
0x17	1111111110001000	16	0x57	1111111110100011	16	0x97	1111111111000101	16	0xD7	1111111111101001	16
0x18	1111111110001001	16	0x58	1111111110100100	16	0x98	1111111111000110	16	0xD8	1111111111101010	16
0x19	1111111110001010	16	0x59	1111111110100101	16	0x99	1111111111000111	16	0xD9	1111111111101011	16
0x1A	1111111110001011	16	0x5A	1111111110100110	16	0x9A	1111111111001000	16	0xDA	1111111111101100	16
0x20	—	—	0x60	—	—	0xA0	—	—	0xE0	—	—
0x21	11010	5	0x61	1111001	7	0xA1	111111000	9	0xE1	11111111100000	14
0x22	11110111	8	0x62	11111110111	11	0xA2	1111111111001001	16	0xE2	1111111111101101	16
0x23	1111110111	10	0x63	1111111110100111	16	0xA3	1111111111001010	16	0xE3	1111111111101110	16
0x24	111111110110	12	0x64	1111111110101000	16	0xA4	1111111111001011	16	0xE4	1111111111101111	16
0x25	111111111000010	15	0x65	1111111110101001	16	0xA5	1111111111001100	16	0xE5	1111111111110000	16
0x26	1111111110001100	16	0x66	1111111110101010	16	0xA6	1111111111001101	16	0xE6	1111111111110001	16
0x27	1111111110001101	16	0x67	1111111110101011	16	0xA7	1111111111001110	16	0xE7	1111111111110010	16
0x28	1111111110001110	16	0x68	1111111110101100	16	0xA8	1111111111001111	16	0xE8	1111111111110011	16
0x29	1111111110001111	16	0x69	1111111110101101	16	0xA9	1111111111010000	16	0xE9	1111111111110100	16
0x2A	1111111110010000	16	0x6A	1111111110101110	16	0xAA	1111111111010001	16	0xEA	1111111111110101	16
0x30	—	—	0x70	—	—	0xB0	—	—	0xF0 (ZRL)	1111111010	10
0x31	11011	5	0x71	1111010	7	0xB1	111111001	9	0xF1	111111111000011	15
0x32	11111000	8	0x72	11111111000	11	0xB2	1111111111010010	16	0xF2	1111111111110110	16
0x33	1111111000	10	0x73	1111111110101111	16	0xB3	1111111111010011	16	0xF3	1111111111110111	16
0x34	111111110111	12	0x74	1111111110110000	16	0xB4	1111111111010100	16	0xF4	1111111111111000	16
0x35	1111111110010001	16	0x75	1111111110110001	16	0xB5	1111111111010101	16	0xF5	1111111111111001	16
0x36	1111111110010010	16	0x76	1111111110110010	16	0xB6	1111111111010110	16	0xF6	1111111111111010	16
0x37	1111111110010011	16	0x77	1111111110110011	16	0xB7	1111111111010111	16	0xF7	1111111111111011	16
0x38	1111111110010100	16	0x78	1111111110110100	16	0xB8	1111111111011000	16	0xF8	1111111111111100	16
0x39	1111111110010101	16	0x79	1111111110110101	16	0xB9	1111111111011001	16	0xF9	1111111111111101	16
0x3A	1111111110010110	16	0x7A	1111111110110110	16	0xBA	1111111111011010	16	0xFA	1111111111111110	16

(b)　色差 AC テーブルの符号割り当て例

● 画像ファイルの基本構成

階層モードを除けば，すべてのJPEGファイルは1個のフレームで構成されます．フレームには1個または複数のスキャンを含みます．スキャンごとにひとかたまりの符号化データが含まれます．スキャンごとの付加情報としてスキャン・ヘッダが付加され，さらにフレームごとの付加情報としてフレーム・ヘッダが付加されます．

画像が単一のコンポーネントの場合（モノクロ）や，インタリーブ・スキャンの場合は，1個のフレームには1個のスキャンだけを含みます．複数コンポーネントの画像をノンインタリーブ・スキャンした場合は，1個のフレームに複数のスキャンを含みます．1回のスキャンの符号化データは，通常は1個の符号化セグメントとして扱われます．

ただし，リスタートを行った場合は，リスタートごと符号化データを分割し，リスタートからリスタートまでの期間の符号化データを1個の符号化セグメントとします．このように符号化セグメントを分けるのは，符号化セグメントの先頭でDC係数の予測値をリセットするためです．

JPEGでは，符号化データも付加情報も，すべてバイト単位のデータとして扱います．ヘッダやテーブルは，すべてバイト単位でフォーマットが規定されています．符号化データは可変長符号を用いるため，本来はバイト単位ではありませんが，エンコーダはビット・ストリームを8ビットずつ区切ってファイルに書き出します．もし，最後に端数が出た場合は，0ビット～7ビットの1を付加して，符号化セグメントの長さがバイト単位になるようにします．

すべてバイト単位のデータとして

コンピュータでは2バイト，4バイトというように値をまとめて扱うほうが効率はよいのだが，そのときLSBが先にくるか，MSBが先にくるかなどの違いがあり，機種依存になってしまいやすい問題がある．バイト単位で扱えば，効率はやや悪いのだが，そのような問題はなくなる．

デコーダはファイルから1バイトずつデータを読み出してビット・ストリームを復元します．

最後に余った1の連続は無視します．

● マーカとゼロ挿入

画像ファイルの始まりと終わり，フレームやスキャン，テーブルの始まりの位置を明確に示すために，マーカと呼ばれる特定の符号が使われます（**表1-7**）．

〈図1-16〉 JPEG交換フォーマット

ベースライン・プロセス，1スキャン，リスタートなしの例を示す．モノクロまたはインタリーブ・スキャンの場合，1スキャンで画像が構成される．

(a) もっとも単純な構成

ベースライン・プロセス，3スキャン，リスタートなしの例を示す．3コンポーネントでノンインタリーブ・スキャンの場合，3スキャンで画像が構成される．

(b) スキャンが複数の場合

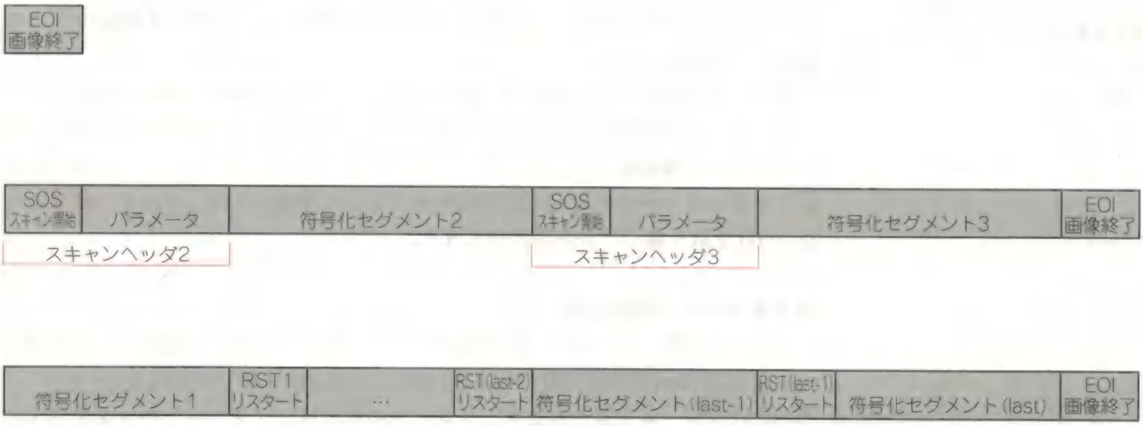
ベースライン・プロセス，1スキャン，リスタートありの例を示す．

(c) リスタートありの場合

〈表 1-7〉JPEG のマーカ

マーカはすべて 2 バイトで，0xFF で始まる．0xFF00 と 0xFFFF はマーカとして使用不可．

符号	名称	意味	備考
●フレーム開始マーカ SOFn（非差分，ハフマン符号化）			
0xFFC0	SOF0	ベースライン	
0xFFC1	SOF1	拡張シーケンシャル DCT	
0xFFC2	SOF2	拡張プログレッシブ DCT	
0xFFC3	SOF3	可逆	
●フレーム開始マーカ SOFn（差分，ハフマン符号化）			
0xFFC5	SOF5	拡張シーケンシャル DCT	
0xFFC6	SOF6	拡張プログレッシブ DCT	
0xFFC7	SOF7	可逆	
●フレーム開始マーカ SOFn（非差分，数値符号化）			
0xFFC8	JPG	将来の拡張のために予約	
0xFFC9	SOF9	拡張シーケンシャル DCT	
0xFFCA	SOF10	拡張プログレッシブ DCT	
0xFFCB	SOF11	可逆	
●フレーム開始マーカ SOFn（差分，数値符号化）			
0xFFCD	SOF13	拡張シーケンシャル DCT	
0xFFCE	SOF14	拡張プログレッシブ DCT	
0xFFCF	SOF15	可逆	
●ハフマンテーブル定義			
0xFFC4	DHT	ハフマン・テーブル定義	
●符号化条件定義			
0xFFCCDAC　数値符号化条件定義			
●リスタート RSTm			
0xFFD0 ～ 0xFFD7	RSTm	リスタート	単独で使用される
●その他			
0xFFD8	SOI	画像開始	単独で使用される
0xFFD9	EOI	画像終了	単独で使用される
0xFFDA	SOS	スキャン開始	
0xFFDB	DQT	量子化テーブル定義	
0xFFDC	DNL	ライン数定義	
0xFFDD	DRI	リスタート間隔定義	
0xFFDE	DHP	階層プログレッション定義	
0xFFDF	EXP		
0xFFE0 ～ 0xFFEF	APPn	アプリケーション用	
0xFFF0 ～ 0xFFFD	JPGn	将来の拡張のために予約	
0xFFFE	COM	コメント	
●予約			
0xFF01	TEM	テンポラリ用	単独で使用される
0xFF02 ～ 0xFFBF	RES	予約	



〈図 1-17〉量子化テーブルとハフマン・テーブル

DQT	Lq	Pq & Tq	Q0	Q1	…	Q63
量子化テーブル定義マーカ	セグメント長	サイズ精度&テーブル指定ラベル	ステップ・サイズ0	ステップ・サイズ1		ステップ・サイズ63
2バイト	2バイト	1バイト	1〜2バイト	1〜2バイト		1〜2バイト

Pqはステップ・サイズの精度を示す4ビットの値Pq=0は8ビット精度，Pq=1は16ビット精度，ベースライン・プロセスではPq=0
Tqはテーブル指定用の4ビットのラベル，フレーム・ヘッダではこのラベルを使ってテーブルを指定する
Tq=0〜3

LqはDQTマーカを除くセグメント長
Lq=2+(1+64)+(1+64)+… または Lq=2+(1+128)+(1+128)+…

複数テーブルの指定時はこの部分を繰り返す

(a) 量子化テーブル定義セグメント

DHT	Lh	Tc & Th	L1	L2	…	L16	V1,1	V1,2	…	V1,L1
ハフマン・テーブル定義マーカ	セグメント長	テーブル種別&指定ラベル	ワード数	ワード数		ワード数	割当値	割当値		割当値
2バイト	2バイト	1バイト	1バイト	1バイト		1バイト	1バイト	1バイト		1バイト

Lnは ビットの可変長ワードの個数
Ln=0ならnビットのワードは使わない

1ビットのワードの割当値(L1個)
Vn,mはLn個中m番目のnビットのワードの割当値

LhはDHTマーカを除くセグメント長
Lh=2+(1+16+ΣLn)+(1+16+ΣLn)+…

複数テーブルの指定時はこの部分を繰り返す

Tcはテーブル種別を示す4ビットのラベル
Tc=0はDCテーブル(または可逆テーブル)，Tc=1はACテーブル
Thはテーブル指定用の4ビットのラベル，スキャンヘッダではこのラベルを使ってテーブルを指定する
ベースライン・プロセスではTh=0〜1

(b) ハフマン定義セグメント

マーカには単独で使われるものと，その後にヘッダなどの付加情報が続くものがあります．マーカと付加情報を合わせたものを**マーカ・セグメント**と呼びます．スキャン・ヘッダやフレーム・ヘッダはマーカ・セグメントです．

マーカ・セグメント

JPEGファイルは，単独のマーカと，マーカ・セグメントと，符号化セグメントの三つの要素から構成されると言える．符号化セグメントは圧縮された画像データ本体，マーカ・セグメントは各種のパラメータ．セグメントという概念と，フレーム，スキャンという概念が重なってややこしくなっている．

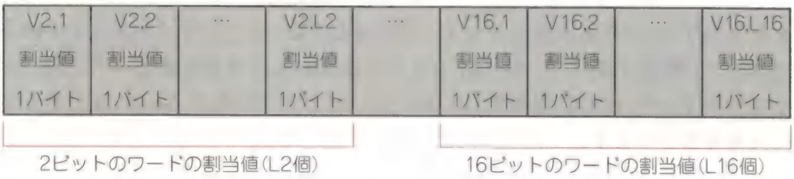
マーカは2バイトの符号で，1バイト目は0xFFです．2バイト目でマーカの種類を示します．デコーダは，画像ファイルの中の0xFFのパターンを検出したら，次のバイトを読んでそれが何のマーカであるかを調べます．

エンコーダはファイルにマーカを書き込むときに，もし必要なら，0xFFを複数回繰り返すことができます．デコーダは，0xFFの次に1個または複数個の0xFFが検出された場合，前方のものはすべて無視して，最後の1個をマーカの開始として認識します．

また，符号化データの中にも0xFFのパターンが出現する場合があります．これをマーカの開始と誤認するのを防ぐため，エンコーダは符号化データの中の0xFFの後に強制的に0x00を挿入します．デコーダでは，0xFFの次が0x00だった場合はマーカではなく符号化データとして認識します．さらに，挿入された0x00を取り除いてから復号化します．

おもなマーカの使い方

JPEGの画像ファイルは，画像開始マーカSOI(0xFFD8)で始まり，画像終了マーカEOI(0xFFD9)で終わります．SOIとEOIは単独で使われるマーカです．その中にフレームが置かれます．フレームの中には，フレーム・ヘッダやスキャン・ヘッダ，量子化テーブルやハフマン・テーブルなどのマーカ・セグメントと，符号化データ・セグメントが置かれます．マーカ・セグメントにはパラメータとしてセグメント長(マーカを除く)が含まれており，セグメントがどこまで続く



かを判断できます(**図 1-16**).

フレームの先頭には，必要があればテーブルやその他の付加情報を置くことができます．続いて，フレーム開始マーカ SOFn (0xFFCn) で始まるフレーム・ヘッダが置かれ，その後に1個または複数のスキャンが置かれます．

フレーム開始マーカには SOF0 (0xFFC0)，SOF1 (0xFFC1)，SOF2 (0xFFC2)，…，SOF15 (0xFFCF) という13個 (SOF4, SOF8, SOF12 は欠けている) があります．SOF0 (0xFFC0) は非差分ハフマン符号化方式のベースライン・プロセス，SOF1 は非差分ハフマン符号化方式の拡張 DCT 利用型シーケンシャル・プロセス，…，というように，マーカの違いで13種類の圧縮/符号化モードを区別できます．

各スキャンの先頭には，必要があればテーブルやその他の付加情報を置くことができます．続いて，スキャン開始マーカ SOS (0xFFDA) で始まるスキャン・ヘッダが置かれ，その後に符号化セグメントが続きます．

量子化テーブルやハフマン・テーブルは，フレーム・ヘッダの前やスキャン・ヘッダの前に置くことができます．各テーブルは，量子化テーブル定義マーカ DQT (0xFFDB)，ハフマン・テーブル定義マーカ DHT (0xFFC4) で始まります (**図 1-17**).

また，マーカの中には APPn (0xFFEn) と呼ばれる**16個の特別のマーカ**があります．これは，アプリケーションが任意に定義して使えるマーカです．APP*n* で始まるマーカ・セグメントも，フレーム・ヘッダやスキャン・ヘッダの前に置くことができます．

16個の特別なマーカ

そのマーカの使い方や，そこから始まるマーカ・セグメントのフォーマットを各アプリケーションが任意に決めることができる．そのとき，たとえば「APP0 マーカの次の1バイトがそのマーカ・セグメントの機能を示す」というように定義すれば，実際には APP0 マーカを256通りに使うことができる．16個の機能しか利用できないわけではない．

フレーム・ヘッダとスキャン・ヘッダ

フレーム・ヘッダには，そのフレーム内に共通のパラメータとして，サンプル精度P，画像のライン数Yと各ラインのサンプル数X，フレーム内のコンポーネント数Nfが置かれます．さらに，各コンポーネントごとに，コンポーネント指定のためのラベルCi，水平サンプリング・ファクタHiと垂直サンプリング・ファクタVi，量子化テーブル指定Tqiというパラメータが繰り返されます(**図1-18**)．

スキャン・ヘッダには，スキャン内のコンポーネント数Nsが置かれます．さらに，各コンポーネントごとに，コンポーネント指定のためのラベルCsj，ハフマン・テーブル指定TdjとTaj，スペクトラル開始Ss，スペクトラル終了Se，サクセッシブ・アプロキシメーション上位Ahと下位Alというパラメータが繰り返されます(**図1-19**)．Ss, Se, Ah, Alの4個はおもに可逆圧縮モードで用いるパラメータであり，DCT利用型ではSs, Ah, Alはいずれも0, Seは63を書くように決められています．

〈図1-18〉フレーム・ヘッダ

SOFn	Lf	P	Y	X	Nf	Ci	Hi & Vi	Tqi
フレーム開始マーカ	セグメント長	サンプル精度	ライン数	ライン内のサンプル数	フレーム内のコンポーネント数	コンポーネント指定ラベル	水平&垂直サンプリング・ファクタ	量子化テーブル指定
2バイト	2バイト	1バイト	2バイト	2バイト	1バイト	1バイト	1バイト	1バイト

LfはSOFnマーカを除く
セグメント長
Lf=8+3×Nf

画像のピクセル数を示す
(最大64K×64K)

コンポーネント数だけ
この部分を繰り返す

ベースライン・プロセスでは
P=8(8ビット精度)

〈図1-19〉スキャン・ヘッダ

SOS	Ls	Ns	Csj	Tdj & Taj	Ss	Se	Ah & Al
スキャン開始マーカ	セグメント長	スキャン内のコンポーネント数	コンポーネント指定	ハフマン・テーブル指定	スペクトラル開始	スペクトラル終了	サクセッシブ・アプロキシメーション
2バイト	2バイト	1バイト	1バイト	1バイト	1バイト	1バイト	1バイト

LsはSOSマーカを除く
セグメント長
Ls=6+2×Ns

コンポーネント数だけ
この部分を繰り返す

Nf=Nsならインタリーブ・スキャン
Nf>Ns=1ならノンインタリーブ・スキャン

DCT利用型ならSs=0，Se=63
ベースライン・プロセスでは
Ah=Al=0

〈図1-20〉JFIFの規定

SOI	APP0	セグメント長	JFIF識別子					JFIF	単位	水平ピクセル密度	垂直ピクセル密度	サムネール情報				
			'J'	'F'	'I'	'F'	'¥0'	バージョン				水平サイズ	垂直サイズ	ピクセル(0,0)	…	ピクセル(Xt, Yt)
0xFFD8	0xFFE0	L	0x4A	0x46	0x49	0x46	0x00	0x102	U	Xd	Yd	Xt	Yt	R G B		R G B

↑ファイル先頭
JPEGのSOIマーカ
JPEGファイルに必須

サムネール不要の場合，およびRGB以外の
サムネールの場合はこの部分なし．
RGB以外のサムネールは拡張APP0マーカ
・セグメントに記述する．

JFIFのAPP0マーカ・セグメント
JFIF準拠のJPEGファイルに必須
セグメント長L=2+2+5+2+1+2+2+1+1+3XtYt=18+3XtYtバイト
単位U=0：単位なし，U=1：ドット/インチ，U=2：ドット/センチ

JPEGでは，パラメータを省略した場合のデフォルト値は規定されていません．必要なパラメータはすべて書き込まなければなりません．しかし，フレーム・ヘッダやスキャン・ヘッダに定義されたパラメータは，復号化のためにかならずしも十分とは言えません．アプリケーションによっては，フレーム・ヘッダやスキャン・ヘッダのパラメータを使わず，独自のAPP*n*マーカを定義して付加情報を交換するものもあります．

JFIFが規定する共通フォーマット

ファイル交換のための共通フォーマットJFIFの規定

つぎに，JPEGファイル交換のための最小限の共通フォーマットとしてJFIF(Jpeg File Interchange Format)の規定を紹介します．これはC-Cube社が定めた仕様であり，JPEGの規格ではありません．しかし，現在流通しているJPEGファイルやソフトウェアの多くはJFIFにも準拠しており，デファクト・スタンダードと見なされています．

JFIF仕様はJPEG規格に準拠しており，JPEGに反する規定はありません．JPEGのもつすべての圧縮/符号化方式が利用可能ですが，データ互換性を高めるためにベースライン・プロセスを用いることを推奨しています．

JPEGの交換フォーマットでは，画像の解像度（ピクセル数）はもちろんヘッダに付加されますが，プリンタや印刷機などでとくに必要とされる画像の実サイズに関する情報が欠けています．また，一般的なベースライン・プロセスでは，検索のためのサムネール画像を元の画像とまとめてファイル化できない点も不便です．これらの問題は，アプリケーションごとにAPP*n*マーカ・セグメントを定義して使えば解決できますが，その部分の互換性がなくなってしまいます．

JFIFは，これらの問題を解決するために，欠けている情報を記述する共通フォーマットを規定したものです．

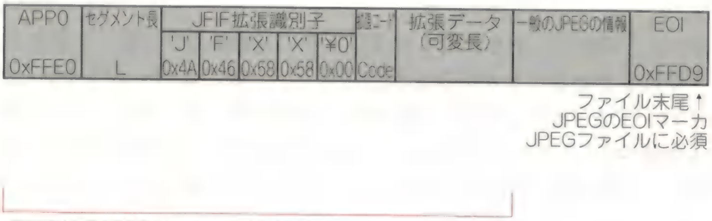
JFIF

JFIFを用いるもう一つの利点は，そのファイルがJPEGフォーマットであることを判定しやすくすることである．JPEGの規格では，ファイルの先頭がSOIマーカ（0xFFD8）であることは決まっているが，その次になにが来るかわからない．アプリケーションとしては，最初の2バイトだけに頼ってファイルの判定をしなければならない．JFIFを用いれば，簡単かつ確実にJPEGファイルを検出できる．

APP0マーカ・セグメント

JFIFはあくまでもJPEGの規格にしたがった形で付加情報を拡張しています．そのため，JPEGにない新しいマーカを追加せず，JPEGでベンダ独自の仕様が認められているAPP0マーカを利用します．JFIFでは，APP0マーカ・セグメントとしてバージョン番号，ピクセル密度（水平/垂直），サムネール画像などの情報を付加します．

また，JFIFはほかのアプリケーションとの干渉をできる限り避けるため，JFIFで使用するAPP0マーカには目印を付けることにしています．それによっ



JFIFの拡張APP0マーカ・セグメント
拡張情報が不要の場合はこの部分なし．
拡張コード Code=0x10：JPEGサムネール，Code=0x11：1バイト/ピクセルのサムネール，Code=0x13：3バイト/ピクセルのサムネール

て，ほかのアプリケーションでも独自のAPP0マーカを利用できます．

JFIFの仕様書には，SOI（画像開始）マーカの直後にAPP0マーカを置くべきことと，APP0マーカから始まるマーカ・セグメントのフォーマットが規定されています（図1-20）．

JFIFのAPP0マーカ・セグメントはSOIマーカの直後に置き，3バイト目から文字列“JFIF”を置かなければなりません．JFIFファイルを復号化/再生するアプリケーションは，ファイル先頭の2バイトがSOI（0xFFD8）で，次の2バイトがAPP0（0xFFE0）であることを検出したら，2バイト（セグメント長）を飛ばして，その次の5バイトが文字列‘J’，‘F’，‘I’，‘F’，‘¥0’（0x4A46494600）であれば，このファイルがJFIF仕様であると判断できます．

また，JFIFではアプリケーションの要求に応じた拡張仕様も規定されており，拡張フォーマットにもAPP0マーカを使います．拡張APP0マーカ・セグメントはJFIFのAPP0マーカ・セグメントの直後に置き，3バイト目から文字列“JFXX”を置かなければなりません．

それ以外の位置にあるAPP0マーカ・セグメントや，3バイト目から“JFIF”または“JFXX”の文字列が現れないAPP0マーカ・セグメントは，JFIFのものではなく，アプリケーションが定義したものと見なすことができます．

JFIFの付加情報

JFIFのAPP0マーカ・セグメントには，文字列“JFIF”に続いてバージョン番号（2バイト），ピクセル密度の単位（1バイト），水平ピクセル密度もしくはアスペクト比（2バイト），垂直ピクセル密度もしくはアスペクト比（2バイト），サムネール画像の水平サイズ（1バイト），サムネール画像の垂直サイズ（1バイト），サムネール画像のRGBデータ（3×水平サイズ×垂直サイズ）が置かれます．

バージョン番号はJFIFのバージョンを示すもので，現行ではVer. 1.02（0x0102）です．

JPEG規格では，画像の解像度（水平ピクセル数×垂直ピクセル数）の情報はフレーム・ヘッダに含まれますが，ピクセル数と画像の実サイズとの関係を示す情報がありません．JFIF仕様ではドット/inchまたはドット/cmで水平(X)と垂直(Y)のピクセル密度をそれぞれ表すことができます．また，水平/垂直のアスペクト比だけを表すこともできます．

インターネット画像のようにディスプレイ表示が前提であれば，ピクセル密度はディスプレイ環境によって決まり，画像の実サイズもそれで決まってしまいます．低解像度の画像の実サイズは小さく，高解像度の画像の実サイズは大きく表示されます．しかし，プリンタなど紙に画像を出力する場合には，ピクセル密度は各画像ごとに変更できます．プリンタの最高解像度の範囲内であれば，同じ解像度（ピクセル数）の画像を小さく出力することも大きく出力することも可能です．

JPEG規格では，階層モードを利用すればフルサイズ画像とサムネール画像を1個のファイルにまとめることができます．しかし，一般的なベースライン・プロセスではそれができません．JFIF仕様では，APP0マーカ・セグメントの中にサムネール画像のRGBデータを埋め込むことができます．RGB以外のサムネール画像を扱いたい場合は，拡張APP0マーカ・セグメントを用います．

バージョン番号

頻繁に機能が追加される規格では，ファイルがその規格のどのバージョンに従って作られたかを示す情報が必要．本来はJPEGの規格にちゃんとファイル・ヘッダが定義されていれば，その中でバージョン番号などを扱うことができる．JFIFにはバージョン番号を示す部分があるが，実際にはこの10年規格は変わらなかった．

第1-2章　静止画データの画像フォーマット

TIFF

宮崎　仁

〈名称〉
Tag Image File Format
〈現行バージョン〉
Rev. 6.0（1992年6月）
〈発行者〉
Adobe Systems Inc.
〈情報入手先〉
http://www. adobe. com/（Adobe社）

周辺機器から取り込んだ画像データを利用するためのファイル・フォーマット

●柔軟で自由度の高い構造をもつ画像ファイル・フォーマット

TIFF（ティフ）（Tag Image File Format）は，静止画データの画像ファイル・フォーマットとしてもっとも古くから使われているものの一つです．ページ記述言語の**PostScript**やDTPソフトウェアの**PageMaker**で知られるAldus社（現在はAdobe社）が，Microsoft社の協力を得て作った規格です．Aldus社が仕様を公開して普及に努めたこともあって，デファクト・スタンダードとして広く使われるようになりました．

もともと，TIFFが作られたのには二つの大きな目的があったと言われています．一つはスキャナなどの周辺機器から取り込んだ画像データをPageMakerで利用すること，もう一つはMacintoshで成功を収めたPageMakerをWindows 2.xに移植することでした．

コンピュータやOS，スキャナやプリンタなどの機器，DTPやペイント，フォト・レタッチなどのアプリケーションは，それぞれ独自の形式で画像データを扱っています．TIFFは，異なる環境，異なる機器，異なるアプリケーション間で画像データを交換するための交換フォーマットとして作られました．そのため，さまざまな独自形式とTIFFの間で容易にフォーマットを変換できるように，TIFFではきわめて柔軟で自由度の高い構造が工夫されています．

TIFFファイルは，さまざまな画像情報を**タグ**（Tag）によって整理して，画像のビット・マップ・データと合わせて一つのファイルにまとめたものです．Tag Image File Formatという名称もここから付けられました．TIFFの仕様書には多数のタグが規定されています．さらに，アプリケーションが独自に機能を追加したり，特定のグループ内だけでローカルに機能を定義するためのプライベート・タグも用意されています．

Adobe社ではローカルTIFF仕様が互いに干渉するのを防ぐため，プライベート・タグの登録を受け付けています．

原画像はラスタ・グラフィック（ビット・マップ）データであれば，モノクロからフル・カラーまで，2値画像から自然画像（連続階調画像）まで幅広くサポート

PostScript
印刷物のデザインやレイアウト，そこで使われている文字や図形などを正確に記述するために作られた言語．1985年に最初のバージョンが発表され，それ以来DTP分野でのデファクト・スタンダードとして普及している．アウトライン・フォントとベクタ・グラフィックスを採用し，高品位の印刷が可能．最近では，電子ドキュメントの標準フォーマットであるPDF（Portable Document Format）や，次世代のWeb画像形式として注目されるSVG（Scalable Vector Graphics）などもPostScriptをベースにしている．

PageMaker
DTP（Desk Top Publishing）ソフトウェアの元祖．それまで紙の上で作業していた印刷物の編集やデザイン，ページ・レイアウトを，すべてパソコン上の作業に移しかえることができる．

タグ
もともとは荷札の意味．コンピュータや通信の分野では，ひとかたまりのデータの中身を簡単に識別できるように付加された短い情報を一般にタグと呼んでいる．

パック・ビット

Macintosh で使われている圧縮方式.

MH

修正ハフマン（Modified Huffman）方式の略. 2 値画像データを効率よく圧縮できる方式で, FAX で使われている.

LZW

さまざまな画像データを高効率に圧縮できる方式として知られており, 圧縮/伸長の処理も高速にできる優れた方式. ところが, GIF に採用されて Web 画像の圧縮に広く使われるようになったため, それまで事実上黙認されていた特許ライセンスの扱いが急に厳しくなってしまった.

プリプレス

文字どおり, 印刷（press）の前処理のことを指す. DTP によってパソコン上で編集されたデータをイメージ・セッタにかけてフィルムを出力し, そのフィルムを製版機にかけて印刷用の刷版を作り, その刷版を印刷機にかけて印刷する, という流れで処理が進められるが, その中の製版のプロセスをプリプレスと呼ぶ. 最近は, DTP データからダイレクトに製版する方法もある.

します. また, 非圧縮およびパック・ビット, MH, LZW, JPEG など各種の圧縮データを扱えます. さらに, TIFF では複数の画像を 1 個のファイルとしてまとめて扱うことができます.

● 規格化の経緯とプライベート・タグを使ったいろいろな TIFF

TIFF の最初のバージョンは 1986 年に発表されました. それ以前に作られていた規格案から通算して, Aldus 社（現 Adobe 社）ではこれを Rev. 3.0 と呼んでいました. そのため, 翌 1987 年の改訂版は Rev. 4.0, さらに 1988 年の改訂版は Rev. 5.0 と名付けられました. 現行バージョンは, 1992 年に発表された Rev. 6.0 です. TIFF の仕様書（pdf ファイル）は Adobe 社の Web サイトで公開されています.

初期の TIFF（Rev. 4.0 以前）はタグの規定に不十分な部分があり, ファイルの互換性に問題があると言われていました. 実際, あるアプリケーションで作った TIFF ファイルをほかのアプリケーションで正常に読み込めなかったり表示できないという問題もいくつか発生しました. しかし, Rev. 5.0 や現行バージョンの Rev. 6.0 では定義が厳密になったため, このような非互換性の問題はなくなってきました.

ただし, TIFF の仕様はオプションの範囲が広く, いくつかの必須のタグを除けば, どのタグをサポートするかはアプリケーションに任されています. ファイルの互換性を高めるには, TIFF ライタ（TIFF ファイルを生成するアプリケーション）は必須のタグだけを使用するようにして, いっぽう TIFF リーダ（TIFF ファイルを解読, 表示するアプリケーション）はなるべく多くのタグをサポートするようにソフトウェアを作らなければなりません.

また, プライベート・タグを使って, 特定用途向けのローカル TIFF 仕様がいくつも作られています. たとえば, プリプレス業界で利用される TIFF/IT, 日本新聞協会が定めた NSK TIFF, 地図や地理的情報を盛り込んだ GeoTIFF, FAX アプリケーションで利用される TIFF クラス F などが知られています. これらのローカル仕様は, 特定の用途かつ特定のグループ内での利用だけを目的としたもので, 不特定多数のユーザ間のデータ互換性はもともと考慮していません. Adobe 社自身も PageMaker 仕様などの拡張 TIFF 仕様を作っています.

さて, TIFF はもともと Macintosh と Windows（AT 互換機）の橋渡しを目的として作られたことから, 異機種間で画像データ交換を容易にできるように考えられていました. しかし, 当時の Windows 2.x はあまり普及せず, DOS では画像データの本格的な利用は難しかったので, AT 互換機のユーザには TIFF はほとんど知られていませんでした.

その後, Windows 3.x や Windows 9x が成功を収めて, AT 互換機でも本格的に画像データが利用できるようになりましたが, Microsoft 社では, Windows 3.x 以降では TIFF の採用を見送り, かわりに DIB（Device Independent Bitmap, いわゆる BMP フォーマット）を標準画像フォーマットとして採用しました.

このため, 1980 年代後半から 1990 年代前半にかけて, TIFF はおもに Macintosh で使われる画像フォーマットと見なされていました. このことは, 後に Windows 9x のユーザやデベロッパにとって TIFF がわかりにくいと言われる一つの理由となっていました.

しかし, 最近ではマルチプラットフォームでのデファクト・スタンダードとして TIFF が利用されています.

ベースラインTIFFと拡張TIFF（Rev. 6.0の仕様）

TIFFの規格は大別するとベースラインTIFFと拡張TIFFに分けられます(図2-1).

ベースラインTIFFはもっとも基本的な方式です．特定用途向けのアプリケーションにはベースラインTIFFをサポートしないものもありますが，一般的なアプリケーションの大部分はベースラインTIFFをサポートしています．ただし，ベースラインTIFFの中で規定された仕様がすべて必須というわけではなく，オプションも数多く含まれています．このことも，TIFFがわかりにくいと言われる原因の一つとなっています．

以前のバージョンでは，2値画像を扱うTIFFクラスB，FAX画像を扱うTIFFクラスF，グレー・スケール（モノクロ連続階調）画像を扱うTIFFクラスG，パレット・カラー画像を扱うTIFFクラスP，RGBフルカラー画像を扱うTIFFクラスRというようにクラス分けされていました．**Rev. 6.0ではクラス分けは廃止され**，ベースラインTIFFの中の2値画像，グレー・スケール画像，パレット・カラー画像，RGBフルカラー画像と呼ばれるようになりました．なお，FAX画像はベースラインTIFFではなく，拡張TIFFに含まれます．

拡張TIFFには，さまざまなTIFFの拡張機能が含まれています．拡張TIFFを利用する場合でも，フォーマットの基本部分はベースラインTIFFに従います．おもな拡張機能としては，G3/G4 FAXで使われている2値符号化（ITU-T T.4/T. 6），**一つの文書に含まれる複数の画像を一つのTIFFファイルで扱う方式**，画像をタイル化して扱う方式，LZW/差分予測/JPEGなどの圧縮方式，CMYK/YCbCr/CIELabなどの画像形式があります．

パレット方式やLZW圧縮のサポートはRev. 5.0から，画像のタイル化やJPEG圧縮，CMYK/YCbCr/CIELabのサポートはRev. 6.0から取り入れられました．

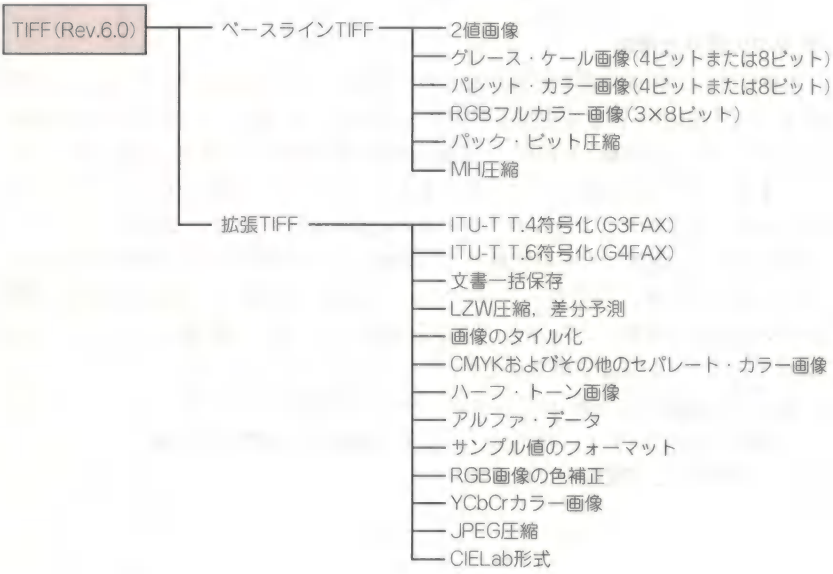
Rev. 6.0ではクラス分けは廃止され

呼び方が変わっただけで，規格の内容が変わったわけではない．TIFFは画像の種類によって必要なタグが変わるので，クラス分け自体はわかりやすいと評判がよかったのだが….

一つの文書に含まれる複数の画像を一つのTIFFファイルで扱う方式

マルチページTIFFなどと呼ばれている．対応しているアプリケーションはまだあまり多くないようだ．

〈図2-1〉TIFFの規格



ベースライン TIFF の構成

必須，強く推奨，推奨，非推奨とレベル分けされている仕様

ベースライン TIFF はもっとも基本的な方式であり，多くのアプリケーションがサポートしています．初期のバージョンから大きな仕様の変化はないので，互換性が高い部分と言えます．ベースライン TIFF で規定している仕様のすべてが必須というわけではなく，必須（すべての TIFF ファイルが満たさなければならない），強く推奨（なるべくみたすべき），推奨（満たすことが望まれる）などのレベルに分かれます．

また，ベースライン TIFF の中にはわずかですが非推奨（一般的な画像データ交換の目的にはなるべく使わない）という仕様もあります．

次に，非推奨を除いた一般的なベースライン TIFF の仕様について説明します．

扱える画像は 2 値画像，グレー・スケール画像，パレット・カラー画像，RGB フルカラー画像

ベースライン TIFF では，2 値画像，グレー・スケール画像，パレット・カラー画像，RGB フルカラー画像の 4 種類の基本的な画像を扱うことができます．そして，それぞれに対して必須のタグが規定されています．

グレー・スケール画像は 4 ビット（16 階調）と 8 ビット（256 階調），パレット・カラー画像も 4 ビット（16 色）と 8 ビット（256 色），RGB フルカラー画像は R，G，B 各 8 ビットの 24 ビット・カラー（1677 万色）を扱えます．それ以外の階調や色数は拡張 TIFF の範囲になります．

パレット・カラーの場合，**パレット情報**は TIFF ファイルに内蔵されます．それによって，表示環境が違っても正しくファイルを表示することができます．

非圧縮画像のほかに，パック・ビット方式と MH（Modified Huffman）方式の圧縮もベースライン TIFF の範囲です．ただし，これらはいずれも必須ではなく，圧縮/伸長アルゴリズムをアプリケーションに実装するかどうかは任意です．パック・ビット方式は，Macintosh で標準として採用されているものです．また，MH 方式は G3/G4 FAX で使用されているものと同じです．

パレット情報

パレット・カラーの場合，パレット番号に対して具体的な色（RGB 情報）を割り当てるカラー・パレットがないと，画像をどんな色で表示すればいいかわからない．そこで，TIFF, DIB, GIF などパレット・カラーを扱う画像ファイルには，カラー・パレットの情報も一緒に入れておく．

タグの書式と種類

TIFF では，画像情報を記述するために多数のタグが規定されています．**タグは 2 バイトのコード**と可変長のパラメータからなります．パラメータには整数（1/2/4 バイト），分数（有理数），実数（浮動小数点数）や文字列（ASCII）などがあります．タグの中にはパラメータをもたないタグや，複数のパラメータをもつタグもあります．どのようなパラメータが必要かはタグによって異なります．

タグ・コードは 2 バイトの整数で，0 x0000 ～ 0 xFFFF（0 ～ 65535）の値をとることができます．現行バージョンでは，0 x00FE（254）～ 0 x0214（532）の範囲と 0 x8298（33432）に 74 個のタグが定義されています（**表 2-1**）．このうち 36 個がベースライン TIFF のタグです．

また，0 x8000（32768）以上のタグ・コードは原則としてプライベート・タグとして開放されています．例外として，0 x8298（33432）だけはベースライン TIFF のタグとして使われています．

タグは 2 バイトのコードと…

Rev. 6.0 では，2 バイトのコードだけをタグと呼び，コードと可変長のパラメータを合わせたものを TIFF フィールドと呼ぶ，というように用語の定義が変更されている．しかし，ここでは Rev. 5.0 までの定義にしたがっており，タグとフィールドという使い分けはしていない．

タグのパラメータは，

① BYTE（符号なし 1 バイト整数）
② ASCII（0 x00 で終端された文字列）
③ SHORT（符号なし 2 バイト整数）
④ LONG（符号なし 4 バイト整数）
⑤ RATIONAL（分子 4 バイト/分母 4 バイトの符号なし分数）
⑥ SBYTE（符号付き 1 バイト整数）
⑦ UNDEFINED（用途を定めない 1 バイト 2 進データ）
⑧ SSHORT（符号付き 2 バイト整数）
⑨ SLONG（符号付き 4 バイト整数）
⑩ SRATIONAL（分子 4 バイト/分母 4 バイトの符号付き分数）
⑪ FLOAT（IEEE の 32 ビット浮動小数点数）
⑫ DOUBLE（IEEE の 64 ビット倍精度浮動小数点数）

があります（**図 2-2**）.

このうち，SBYTE から後のものは Rev. 6.0 で追加された新しい仕様です．**表 2-1** に示した TIFF の標準のタグ（ベースライン TIFF，拡張 TIFF）では，これらの新しいパラメータは使われていません．

● TIFF ファイルの構成

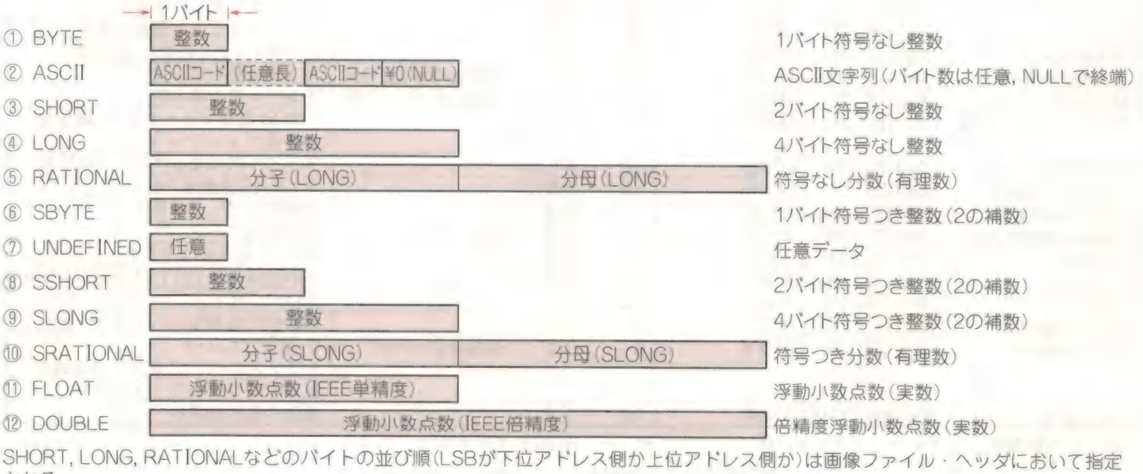
TIFF の画像ファイルは大きく分けて画像ファイル・ヘッダ，画像ファイル・ディレクトリ（IFD），ビット・マップ・データによって構成されます（**図 2-3**）．

画像ファイル・ヘッダはかならずファイルの先頭に置かれますが，画像ファイル・ディレクトリやビット・マップ・データはファイル内の任意の場所に置くことができます．ファイルに複数の画像が含まれる場合は，画像ファイル・ディレクトリやビット・マップ・データも複数になります．また，1 個の画像のビット・マップ・データが複数に分割されている場合もあります．

このように，TIFF ファイルのデータ構造は 1 列のデータ・ストリームではなく，**いくつかのブロックを集めた構造**になっています．画像ファイル・ヘッダ以外のブロックの順序は任意です．ブロックとブロックの間に空き（ファイルの未使用部分）が存在する場合もあります．

いくつかのブロックを集めた構造
これは TIFF の大きな特徴と言えるだろう．画像フォーマットに限らず，ほかの多くのフォーマットは，基本的には 1 列のデータ・ストリームを頭から順番に処理していく．データ・ストリームがブロックに分かれているとしても，一つのブロックが終わった後に次のブロックが続く，というようにシーケンシャルに並んでいる．それに対して，TIFF ではブロックが独立していて，階層的に処理することが可能．

〈図 2-2〉 タグのパラメータ



SHORT, LONG, RATIONALなどのバイトの並び順（LSBが下位アドレス側か上位アドレス側か）は画像ファイル・ヘッダにおいて指定される

〈表 2-1〉 タグの種類と書式

名称	コード		パラメータ型	適用(注)	パラメータ長	パラメータ値
	16 進	10 進				
ImageWidth	0x0100	256	SHORT or LONG	All 必須	1	画像の 1 行のピクセル数
ImageLength	0x0101	257	SHORT or LONG	All 必須	1	画像の行数
BitsPerSample	0x0102	258	SHORT	B 不要	1	1: 白黒 2 値(既定値)
				G/P 必須	1	4: 16 階調(G), 16 色(P) 8: 256 階調(G), 256 色(P)
				R 必須	SamplesPerPixel の値	8, 8, 8: 24 ビット・カラー
Compression	0x0103	259	SHORT	B 必須	1	1: 無圧縮 (既定値) 2: MH 圧縮 32773: パック・ビット圧縮
				G/P/R 必須	1	1: 無圧縮(既定値) 32773: パック・ビット圧縮
PhotometoricInterpretation	0x0106	262	SHORT	B/G 必須	1	0: '0' = 白 1: '0' = 黒
				P 必須	1	3 :パレット・カラー
				R 必須	1	2: RGB
StripOffsets	0x0111	273	SHORT or LONG	All 必須	画像のストリップ数	ストリップのビット・マップ・データの位置
SamplesPerPixel	0x0115	277	SHORT	B/G/P 不要	1	1(既定値)
				R 必須	1	3 またはそれ以上
RowsPerStrip	0x0116	278	SHORT or LONG	All 必須	1	画像の 1 ストリップの行数
StripByteCounts	0x0117	279	LONG or SHORT	All 必須	画像のストリップ数	画像の 1 ストリップのバイト数
XResolution	0x011A	282	RATIONAL	All 必須	1	水平方向解像度
YResolution	0x011B	283	RATIONAL	All 必須	1	垂直方向解像度
ResolutionUnit	0x0128	296	SHORT	All 必須	1	1: 無単位 2: inch(既定値) 3: cm
ColorMap	0x0140	320	SHORT	B/G/P 不要	0	
				P 必須	3 × 16 or 3 × 256	パレットの RGB データ

(注) B: 2 値画像, G: グレー・スケール, P: パレット・カラー, R: RGB フルカラー

(a) 必須のベースライン TIFF タグ

名称	コード		パラメータ型	適用(注)	パラメータ長	パラメータ値
	16 進	10 進				
NewSubfileType	0x00FE	254	LONG	All 任意	1	SubfileType の拡張, ビット 0 ~ 2 がフラグ
SubfileType	0x00FF	255	SHORT	All 任意	1	1: サブファイルはフル解像度画像データ 2: サブファイルは低解像度画像データ 3: マルチページ画像
PhotometoricInterpretation	0x0106	262	SHORT	任意	1	4: 透過マスク
Threshholding	0x0107	263	SHORT	B/G 任意	1	1: ディザ/ハーフトーン不使用(既定値) 2: 規則的なディザ/ハーフトーン使用 3: ランダムなディザ/ハーフトーン使用
CellWidth	0x0108	264	SHORT	B/G 任意	1	規則的なディザ/ハーフトーンのセル幅
CellLength	0x0109	265	SHORT	B/G 任意	1	規則的なディザ/ハーフトーンのセル長
FillOrder	0x010A	266	SHORT	All 任意	1	1: データ左詰め(既定値) 2: データ右詰め
ImageDescription	0x010E	270	ASCII	All 任意		画像の表題
Make	0x010F	271	ASCII	All 任意		スキャナなど入力機器のメーカ
Model	0x0110	272	ASCII	All 任意		スキャナなど入力機器の名称/型番
Orientation	0x0112	274	SHORT	All 任意	1	1: 0 行が上, 0 列が左(既定値)
MinSampleValue	0x0118	280	SHORT	All 任意	SamplesPerPixel の値	サンプル値の最小値
MaxSampleValue	0x0119	281	SHORT	All 任意	SamplesPerPixel の値	サンプル値の最大値
PlanarConfiguration	0x011C	284	SHORT	R 任意	1	1: RGB 混在フォーマット(既定値) 2: 分離フォーマット
FreeOffsets	0x0120	288	LONG	All 非推奨		ファイル内未使用ブロックの位置
FreeByteCounts	0x0121	289	LONG	All 非推奨		ファイル内未使用ブロックのバイト数
GrayResponseUnit	0x0122	290	SHORT	G 任意	1	階調特性データの単位
GrayResponseCurve	0x0123	291	SHORT	G 任意	2^4 or 2^8	階調特性データ
Software	0x0131	305	ASCII	All 任意		画像の作成環境(アプリケーション名)
DateTime	0x0132	306	ASCII	All 任意	20	画像の作成日時(YYYY:MM:DD:HH:MM:SS)
Artist	0x013B	315	ASCII	All 任意		画像の作成者名
HostComputer	0x013C	316	ASCII	All 任意		画像の作成環境 (ハード/OS 名)
ExtraSamples	0x0152	338	BYTE	R 任意	SamplesPerPixel − 3	0: 規定されない 1: アソシエート・アルファ(不透過率) 2: 非アソシエート・アルファ
Copyright	0x8298	33432	ASCII	All 任意		画像の著作権表示

(注) B: 2 値画像, G: グレー・スケール, P: パレット・カラー, R: RGB フルカラー

(b) 任意および非推奨のベースライン TIFF タグ

〈表2-1〉タグの種類と書式(つづき)

名称	コード 16進	コード 10進	パラメータ型	適用(注)	パラメータ長	パラメータ値
BitsPerSample	0x0102	258	SHORT	Sep 拡張	SamplesPerPixelの値	8, 8, 8, 8: CMYKカラー 8, 8, 8: YCbCrカラー
Compression	0x0103	259	SHORT	拡張	1	3: G3FAX(CCITT T.4) 4: G4FAX(CCITT T.6) 5: LZW圧縮　6: JPEG圧縮
PhotometoricInterpretation	0x0106	262	SHORT	拡張	1	5: セパレート・カラー(CMYKなど) 6: YCbCr　7: 1976CIELab
DocumentName	0x010D	269	ASCII	文書 拡張		文書の表題
Orientation	0x0112	274	SHORT	All 拡張	1	2: 0行が上, 0列が右 3: 0行が下, 0列が右 4: 0行が下, 0列が左 5: 0行が左, 0列が上 6: 0行が右, 0列が上 7: 0行が右, 0列が下 8: 0行が左, 0列が下
SamplesPerPixel	0x0115	277	SHORT	Sep 拡張	1	3またはそれ以上, CMYKは4
PageName	0x011D	285	ASCII	文書 拡張		ページの表題
XPosition	0x011E	286	RATIONAL	文書 拡張		ページ左端を基準にした画像左端の位置
YPosition	0x011F	287	RATIONAL	文書 拡張		ページ上端を基準にした画像上端の位置
T4Options	0x0124	292	LONG	G3FAX 拡張	1	ビット0～2がオプション・フラグ
T6Options	0x0125	293	LONG	G4FAX 拡張	1	ビット0～1がオプション・フラグ
PageNumber	0x0129	297	SHORT	文書 拡張	2	ページ番号, 全ページ数
TransferFunction	0x012D	301	SHORT	拡張	1 or 3×16 or 3×256	
Predictor	0x013D	317	SHORT	LZW 拡張	1	差分予測値
WhitePoint	0x013E	318	RATIONAL	R 拡張	2	白x, 白y
PrimaryChromaticities	0x013F	319	RATIONAL	R 拡張	6	3原色(赤x, 赤y, 緑x, 緑y, 青x, 青y)
HalftoneHints	0x0141	321	SHORT	G 拡張	2	ハイライト・レベル, シャドウ・レベル
TileWidth	0x0142	322	SHORT or LONG	Tile 拡張	1	タイル1個の幅(ピクセル数)
TileLength	0x0143	323	SHORT or LONG	Tile 拡張	1	タイル1個の長さ(ピクセル数)
TileOffsets	0x0144	324	LONG	Tile 拡張	1	タイルのビットマップ・データの位置
TileByteCounts	0x0145	325	SHORT or LONG	Tile 拡張	1	タイル1個のバイト数
InkSet	0x014C	332	SHORT	Sep 拡張	1	1: CMYK 2: 非CMYK
InkNames	0x014D	333	ASCII	Sep 拡張		セパレート・カラーの定義(インク名を列挙)
NumberOfInks	0x014E	334	SHORT	Sep 拡張	1	インクの色数
DotRange	0x0150	336	BYTE or SHORT	Sep 拡張	2 or 2×インク数	0%ドットの値, 100%ドットの値
TargetPrinter	0x0151	337	ASCII	Sep 拡張		このセパレート・カラーを使用するプリンタ名
ExtraSamples	0x0152	338	BYTE	All 拡張	追加コンポーネント数	0: 規定されない 1: アソシエート・アルファ(不透過率) 2: 非アソシエート・アルファ
SampleFormat	0x0153	339	SHORT	All 拡張	SamplesPerPixelの値	1: 符号なし整数(既定値) 2: 2の補数形式の符号付き整数 3: IEEE浮動小数点数 4: 定義されていないフォーマット
SMinSampleValue	0x0154	340	Any	All 拡張	SamplesPerPixelの値	MinSampleValueの拡張
SMaxSampleValue	0x0155	341	Any	All 拡張	SamplesPerPixelの値	MaxSampleValueの拡張
TransferRange	0x0156	342	SHORT	拡張	6	
JPEGProc	0x0200	512	SHORT	JPEG 拡張	1	1: ベースライン・プロセス 14: 可逆プロセス(ハフマン符号化)
JPEGInterchangeFormat	0x0201	513	LONG	JPEG 拡張	1	0: JPEGビット・ストリームなし 有効オフセット値: SOIマーカの位置
JPEGInterchangeFormatLength	0x0202	514	LONG	JPEG 拡張	1	0: JPEGビット・ストリームなし 1以上: ビット・ストリームのバイト数
JPEGRestartInterval	0x0203	515	SHORT	JPEG 拡張	1	0: リスタート・マーカなし 1以上: リスタート・インターバル
JPEGLosslessPredictors	0x0205	517	SHORT	JPEG 拡張	SamplesPerPixelの値	不可逆圧縮の予測値選択
JPEGPointTransforms	0x0206	518	SHORT	JPEG 拡張	SamplesPerPixelの値	不可逆圧縮の点変換値
JPEGQTables	0x0207	519	LONG	JPEG 拡張	SamplesPerPixelの値	量子化テーブルの位置
JPEGDCTables	0x0208	520	LONG	JPEG 拡張	SamplesPerPixelの値	DCテーブル/可逆ハフマン・テーブルの位置
JPEGACTables	0x0209	521	LONG	JPEG 拡張	SamplesPerPixelの値	ACテーブルの位置
YCbCrCoefficients	0x0211	529	RATIONAL	Y 拡張	3	RGB-YCbCr変換係数
YCbCrSubSampling	0x0212	530	SHORT	Y 拡張	2	1: 1　2: 1/2 (既定値) 4: 1/4
YCbCrPositioning	0x0213	531	SHORT	Y 拡張	1	輝度サンプルを基準にした色サンプルの位置
ReferenceBlackWhite	0x0214	532	LONG	拡張	2×SamplesPerPixel	

(注) G: グレー・スケール, R: RGBフルカラー, Sep: セパレート・カラー(CMYKなど), Y: YCbCr

(c) 拡張TIFFタグ

TIFF ファイル内部では，オフセット（ファイル先頭を 0 とした相対アドレス）によってブロックの位置を指定します．IFD やビット・マップ・データは，指定されたオフセットをたどって読むことができます．オフセットは 4 バイトなので，TIFF ファイルのサイズは $2^{32}=4$ G バイト以内に制限されます．なお，オフセットの値はワード境界（偶数値）になるようにします．

画像ファイル・ヘッダはファイル全体の情報を示すもので，このファイルが TIFF ファイルであることと，バイトの並び順が**ビッグ・エンディアン**（モトローラ風）か，**リトル・エンディアン**（インテル風）かがわかります．この，バイトの並び順を指定できることは，異機種間でのファイル交換をやりやすくしています．

IFD は，画像に関するさまざまな情報を示すタグを書き並べたものです．TIFF ファイルには少なくとも 1 個の IFD が必要です．原則として，1 個の IFD は 1 個の画像データに対応します．TIFF では複数の画像を一つのファイルにまとめることができますが，その場合は IFD も複数になります．画像ファイル・ヘッダの末尾に 1 個目の IFD のオフセットが書いてあり，1 個目の IFD の末尾に 2 個目の IFD のオフセットが書いてあり，2 個目の IFD の末尾に 3 個目の IFD のオフセットが書いてあり，というように，画像ファイル・ヘッダから出発して次々に IFD のチェーンをたどっていくことができます．

ただしベースライン TIFF の範囲では，ソフトウェアは 1 個の IFD を処理することだけが必須です．2 個目以降の IFD は無視してもかまいません．

TIFF では，1 個の画像のビット・マップ・データを帯状の**ストリップ**に分けて管理します（**図 2-4**）．これは，TIFF がもともとスキャナ・データの取り込みを目的として作られたなごりといえるでしょう．スキャナなどのアプリケーションでは，1 回のスキャンで横方向の 1 行（または複数行）分のデータ（ストリップ）を読み取り，それを繰り返して全画面分のデータを読み取る方式が広く使われています．帯の太さ（ピクセル数）は任意に選ぶことができます．帯の太さを画像の長さと同じに選べば，画像全体を 1 本のストリップとみなして扱うことも可能です．

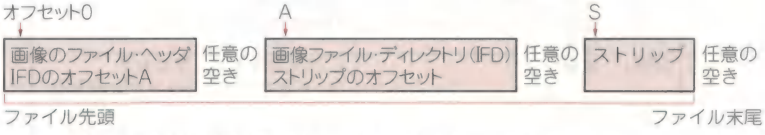
ストリップのオフセットは IFD の中に書いてあり，IFD からたどってビット・マップ・データを読み出すことができます．

ビッグ・エンディアン
16 ビットや 32 ビットのデータをバイト・アドレス空間に配置するとき，MSB から順番に並べる方式．ビッグ・エンディアンという名称は，ガリバー旅行記に出てくる，ゆで卵をかならず大きい方の端から食べる主義の人々から付けられたと言われている．

リトル・エンディアン
16 ビットや 32 ビットのデータをバイト・アドレス空間に配置するとき，LSB から順番に並べる方式．リトル・エンディアンという名称は，ガリバー旅行記に出てくる，ゆで卵を必ず小さい方の端から食べる主義の人々から付けられたと言われている．

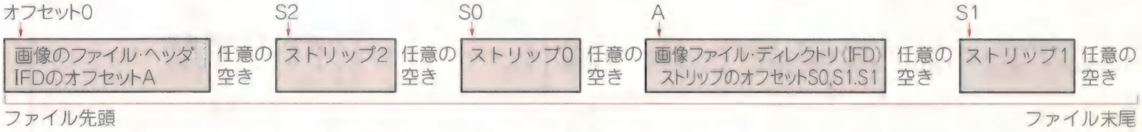
ストリップ
紙や布を細長く引き裂いたものを strip という．また，土地などを細長く区切ったものも strip といい，滑走路や道路を strip と呼ぶこともある．strap（ひも）と間違えやすいので注意．

〈図 2-3〉TIFF ファイルの構成の例



画像ファイル・ヘッダはかならずファイル先頭（オフセット0）に置かれる．
IFDとストリップの順序は任意，空きはなくてもよい．
IFDのオフセットは画像ファイル・ヘッダの後半4バイトで指定する．
ストリップのオフセットは，IFDの中のStripOffsetsタグで指定する

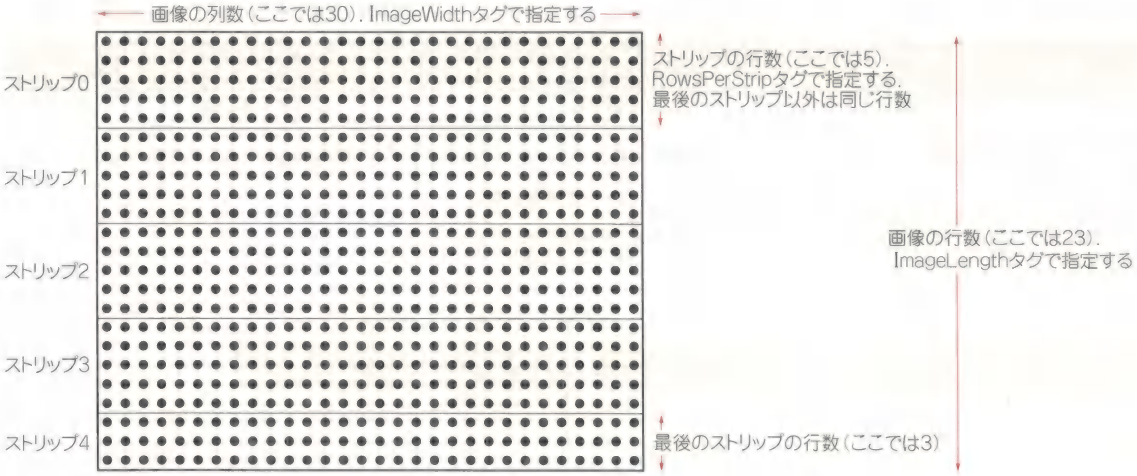
(a) 最小構成──画像ファイル・ヘッダ，1個のIFD,1個のストリップ（ビット・マップ・データ）の例



1個の画像のビット・マップ・データを，3個のストリップに分割して扱っている．
IFDと各ストリップの順序は任意，空きはなくてもよい

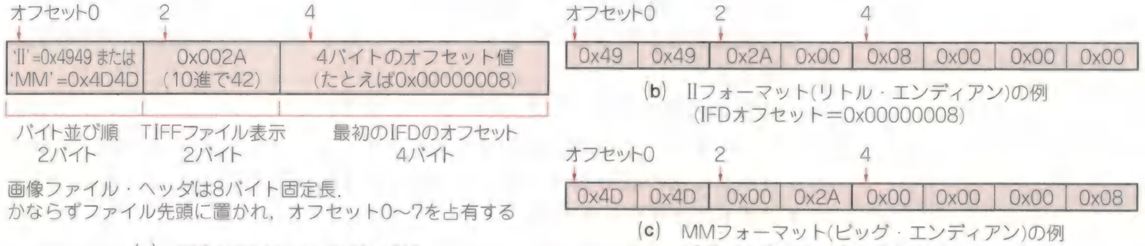
(b) 画像ファイル・ヘッダ，1個のIFD, 3個のストリップの例

〈図 2-4〉ビット・マップとストリップ



30×23ピクセルの画像の列. ●が1個のピクセルを示す.
TIFFファイルでは, ビット・マップ・データをストリップ（細帯）に分割して扱う. 拡張TIFFではタイル分割も可.
ストリップの行数は任意（1～画像の行数）.
ストリップの行数＝1のとき, 1行が1ストリップ
ストリップに分割せず画像全体を1ブロックとして扱う場合, ストリップの行数＝画像の行数とする.
画像に含まれるストリップ数, 最後のストリップの行数は, TIFFァイルでは指定しない. 他のパラメータからの計算で求める.
画像の行数÷ストリップの行数を計算すれば, 商（整数）＋1が画像に含まれるストリップ数, 余りが最後のストリップ行数

〈図 2-5〉画像ファイル・ヘッダ



(a)　画像ファイル・ヘッダの構成

(b)　IIフォーマット（リトル・エンディアン）の例 （IFDオフセット＝0x00000008）

(c)　MMフォーマット（ビッグ・エンディアン）の例 （IFDオフセット＝0x00000008）

● TIFF ファイルの先頭におかれる画像ファイル・ヘッダ

TIFF ファイルはかならず画像ファイル・ヘッダから始まります. 画像ファイル・ヘッダは TIFF ファイルの先頭におかれる 8 バイトのヘッダで, 三つの部分からなります（図 2-5）.

まず, **最初の 2 バイトでバイトの並び順（order）を示し**, 次の 2 バイトでファイルが TIFF 仕様であることを示します.

最初の 2 バイトが文字 "II"（0 x4949）ならバイト並びはリトル・エンディアン（2/4 バイト・データは LSB から順に並ぶ）, 文字 "MM"（0 x4D4D）ならバイト並びはビッグ・エンディアン（2/4 バイトデータは MSB から順に並ぶ）です. 次の 2 バイトはかならず整数値 42（0 x002A）がおかれます. この最初の 4 バイトを見れば, ファイルが TIFF 仕様であることを認識できます.

さらに, **画像ファイル・ヘッダの後半の 4 バイト**は 1 個目の IFD のオフセットを示します. IFD は画像ファイル・ヘッダの後に続けておくこともできますし, 離れた場所におくこともできます.

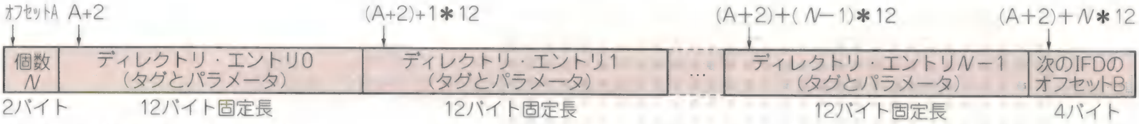
最初の 2 バイトでバイトの並び順（order）を示し

TIFF ファイルでは 16 ビットや 32 ビットのパラメータがたくさん使われている. TIFF ファイルを読むときは, まずバイトの並び順をしっかり認識しておかないと, その後の 16 ビットや 32 ビットのパラメータが正しく読めない.

画像ファイル・ヘッダの後半の 4 バイト

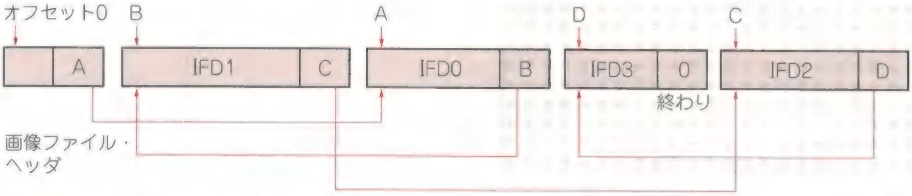
この 4 バイトは IFD のオフセットを示す 32 ビットの数値である. 指定されたバイトの並び順（order）に従って読まないと, IFD が見つからない.

〈図 2-6〉 画像ファイル・ディレクトリ (IFD)



オフセットAから始まるIFDを示す.
Nは，このIFDに含まれるディレクトリ・エントリの個数．1個のディレクトリ・エントリに1個のタグを記述するので，N はタグの個数と一致する.
ディレクトリ・エントリはタグとパラメータの情報．タグ・コードの昇順に並ぶ.
Bは次のIFDのオフセット．次のIFDがない場合はB=0が入る.
IFDにはこれらの情報が順序通りに並んでいる．IFDの長さは(2+N＊12+4)バイト

(a) IFDの構成



画像ファイル・ヘッダから出発して，次々にIFDをたどっていくことができる.
次のIFDのオフセット=0になったら，そこで終わり.
ただし，複数のIFDを処理する機能はオプション

(b) IFDのチェーン

複数の IFD を処理する機能はオプションです

ベース・ライン TIFF では，TIFF リーダは原則としてすべてのオプション機能をサポートすることが必須となっている．しかし，この複数 IFD の処理だけは例外で，TIFF リーダはサポートしなくても良いことになっている．そのため，マルチページ TIFF のファイルを未対応のアプリケーションに読み込むと，一つ目の画像だけは正常に処理できるが，二つ目以降の画像は無視されてしまう.

● IFD (画像ファイル・ディレクトリ) とディレクトリ・エントリ

IFD は，ディレクトリ･エントリと呼ばれる 12 バイト固定長の情報部を連ねたものです (**図 2-6**)．このディレクトリ･エントリには，タグで表されるさまざまな画像情報が書き込まれます．ディレクトリ･エントリの個数はタグの個数と一致します.

IFD の最初の 2 バイトは，その IFD に含まれるディレクトリ･エントリの個数 N を示します．その後に 12 バイト×N のディレクトリ･エントリが続き，最後の 4 バイトに次の IFD のオフセットが置かれます．次の IFD がなければ，最後の 4 バイトには 0 を入れます.

ファイルに複数の画像を含む場合には，画像ファイル･ヘッダには最初の IFD のオフセットが置かれます．最初の IFD の最後の 4 バイトが 2 番目の IFD のオフセット，2 番目の IFD の最後の 4 バイトが 3 番目の IFD のオフセット，…，というように，いくらでも IFD のチェーンを増やすことができます.

ただし，**複数の IFD を処理する機能はオプション**です．TIFF リーダ (TIFF ファイルを読み込む機能をもつアプリケーション) は，画像ファイル･ヘッダに続く最初の IFD は必須として処理しなければなりませんが，2 個目以降は無視することができます.

ディレクトリ･エントリは，タグとパラメータの情報からなります (**図 2-7**)．ディレクトリ･エントリの最初の 2 バイトはタグ･コード，次の 2 バイトはパラメータの型 (type)，次の 4 バイトはパラメータ長 (count)，最後の 4 バイトはパラメータまたはオフセットです．タグのパラメータが 4 バイト以内のときは，ディレクトリ･エントリにはパラメータそのものが書き込まれます．パラメータが 4 バイトを超えるときは，パラメータはファイル内の別の場所に書き，ディレクトリ･エントリにはそのオフセットが書き込まれます.

パラメータの型 (type) は**図 2-2** のように 12 種類が定義されていますが，おもに使われているのは BYTE, ASCII, SHORT, LONG, RATIONAL の 5 種類

〈図 2-7〉ディレクトリ・エントリ

オフセット(A+2)+(i-1)＊12

Tag タグ・コード	type パラメータの種類	count パラメータ長(または個数)	value パラメータ
2バイト	2バイト	4バイト	4バイト

オフセットAから始まるIFDの中の，i 番目のディレクトリ・エントリを示す．
タグ・コードは**表2-1**参照．ディレクトリ・エントリはタグ・コードの小さいものから順に並ぶ(昇順)．
パラメータの種類は**図2-2**参照．現在は1(0x0001)～12(0x000C)の範囲しか使われていないが，将来は拡張の可能性がある．
パラメータ長は，typeで定義されたパラメータの数を示す．type=1(BYTE)でcount=1なら1バイトのパラメータがある．type=4(LONG)でcount=1なら4バイトのパラメータがある．
パラメータのバイト数が4バイト以内なら，ディレクトリ・エントリの最後の4バイトにその値が入る．たとえばtype=1(BYTE)ならcount≦4のとき，type=4(LONG)ならcount≦1のとき

(a)　パラメータが4バイト以内の場合

オフセット(A+2)+(i-1)＊12

Tag タグ・コード	type パラメータの種類 0x0002	count パラメータ長(または個数) 0x00000004	offset オフセット B
2バイト	2バイト	4バイト	4バイト

オフセットB

文字0 'J' 0x4A	文字1 'K' 0x4B	文字2 'L' 0x4C	文字3 'M' 0x4D	文字4 NULL 0x00
1バイト	1バイト	1バイト	1バイト	1バイト

パラメータのバイト数が4バイトを超える場合，IFDの外側にパラメータを置き，ディレクトリ・エントリの最後の4バイトはそのオフセット値が入る．
例として，パラメータが文字列“JKLM”の場合を示す．type=2(ASCII)，count=4である．4文字に終端の0x00(NULL)が付くので，パラメータは5バイトになる

(b)　パラメータが4バイトを超える場合(1)

オフセット(A+2)+(i-1)＊12

Tag タグ・コード	type パラメータの種類 0x0004	count パラメータ長(または個数) 0x00000003	offset オフセット B
2バイト	2バイト	4バイト	4バイト

オフセットB

ストリップ0のオフセット S0	ストリップ1のオフセット S1	ストリップ2のオフセット S2
4バイト	4バイト	4バイト

例として，パラメータが3個のストリップのオフセットの場合を示す．type=4(LONG)，count=3である

(c)　パラメータが4バイトを超える場合(2)

です．

パラメータ長（count）はパラメータの長さもしくは個数を示すものです．このパラメータ長の値は，型（type）で定義されたデータの数を表します．たとえば，BYTE 型（1 バイト整数）でパラメータ長が 3 ならデータは 1× 3＝3 バイト，SHORT 型（2 バイト整数）でパラメータ長が 3 ならデータは 2× 3＝6 バイト，LONG 型（4 バイト整数）でパラメータ長が 3 ならデータは 4× 3＝12 バイトとなります．また，パラメータの型が ASCII の場合，文字列の最後に NULL コード（0 x00）が付加されるので，パラメータ長＋1 がバイト数となります．

● 画像データはビット・マップ・データ

TIFF では，画像データの本体（ビット・マップ・データ）を**ファイル内の任意の場所に**置くことができます．また，データ全体をひとまとめにして扱うこともできますし，適当なサイズのストリップに分割して扱うこともできます．**表 2-1** (a)に示したベースラスン TIFF の必須のタグの中に，これらの情報を記述するためのタグが用意されています．

図 2-4 のように，画像全体のサイズ（ピクセル数）は，ImageWidth（画像の列数），ImageLength（画像の行数）という二つのタグで指定します．いずれも SHORT（2 バイト）または LONG（4 バイト）を用いることができます．SHORT の場合は最大 64 K ピクセル，LONG の場合は最大 4 G ピクセルを指定できます．ただし，**実際には TIFF ファイルのサイズの上限**が 4 G バイトのため，画像データのサイズもそれ以内に制限されます．

TIFF では原則として画像データをストリップ単位に分割して扱います．ストリップは，列数は ImageWidth いっぱい，行数は RowsPerStrip（ストリップの

ファイル内の任意の場所に

画像ファイル・ヘッダだけは常にファイル先頭に置かれるので別だが，それより後ならどこに画像データを置いてもかまわない．画像ファイル・ヘッダに続けて画像データを置き，属性情報（タグ）は後にまとめておく，ということも可能．

実際には TIFF ファイルのサイズの上限

TIFF ファイルは，4 バイトのオフセットでアドレスを指定できる大きさでなければならない．したがって，ファイル・サイズの上限は 2^{32} ＝ 4 G バイトとなる．

行数）タグで指定される帯状の領域です．RowsPerStrip は 1 ～ ImageLength の任意の値を選べます．ストリップの行数は均一ですが，画像の行数がストリップの行数で割り切れない場合，最後のストリップだけ行数が少なくなります．

画像データを何本のストリップに分割するかを指定するタグはありません．アプリケーション側で ImageLength ÷ RowsPerStrip を計算すれば，商（整数）＋1 がストリップ数，余りが最後のストリップの行数になります．また，とくに RowsPerStrip＝ImageLength に選べば，画像全体を分割せずに一つのストリップとして扱うことができます．

図 2-3 のように，各ストリップのデータはファイル内の任意の場所に置くことができます．ストリップのオフセットは StripOffsets タグで指定します．**図 2-7** のように，1 個の StripOffsets タグのパラメータとしてストリップ数分のオフセット値（LONG）を並べることができます．

StripByteCounts タグを用いて，各ストリップのデータのバイト数を指定します．各ストリップに含まれるピクセル数は，ImageWidth と RowsPerStrip によって決まりますが，それが何バイトのデータになるかは画像の種類によって変わります．とくに，画像を圧縮した場合のビット・マップ・データの長さは不定ですが，アプリケーション側ではこの StripByteCounts を参照して必要なバイト数分のデータをファイルから確実に読み取ることができます．

● 画像の種類とビット・マップ・データ

2 値画像では 1 ピクセルのデータは 1 ビットで表すことができます．それに対して，グレー・スケール，パレット・カラー，RGB カラーなどの画像では，1 ピクセルを複数のビットで表します．とくに，RGB カラーの場合は，1 ピクセルに対して赤(R)，緑(G)，青(B)の 3 原色の情報が必要です．**表 2-1** のように，TIFF にはそれらを指定するためのタグも用意されています．

PhotometricInterpretation タグは画像の種類を表すものです．ベースライン TIFF で必須とされている PhotometricInterpretation の値は，0（**'0'＝白のモノクロ画像**），1（**'0'＝黒のモノクロ画像**），2（RGB カラー），3（パレット・カラー）の 4 種類です．PhotometricInterpretation タグでは，2 値画像とグレー・スケールはどちらもモノクロとして共通に扱います．

SamplesPerPixel タグは，画像の 1 ピクセルが何個のサンプル・データから構成されるかを示します．2 値画像，グレー・スケール，パレット・カラーは SamplesPerPixel＝1 です．RGB カラーは少なくとも R，G，B の 3 個のサンプル・データが必要なので，SamplesPerPixel ≧ 3 となります．RGB の SamplesPerPixel は通常は 3 ですが，画像の透明度（不透明度）などを示す追加データがあるときは 4 以上の値をとります．

BitsPerSample タグは，サンプル・データのビット数を示します．2 値画像では 1 個のサンプル・データは 1 ビット，グレー・スケールとパレット・カラーでは 4 ビット（16 階調，16 色）または 8 ビット（256 階調，256 色）です．RGB カラーは R，G，B それぞれ 8 ビットで，BitsPerSample タグのパラメータも 8，8，8 と各サンプル・データごとに分けて書きます．

RGB カラーでは一般に，**RGB のプレーン**（平面）を分離して扱う方式（Planar Format）と分離しない方式（**Chunky Format**）の二つがあります（**図 2-8**）．プレーンを分離する方式では，各プレーンは 8 ビットのグレー・スケールと同様のデータ並びになります．一方，プレーンを分離しない方式では，1 枚のプレーンの中に RGB のデータが混在します．TIFF では，このプレーンを分離

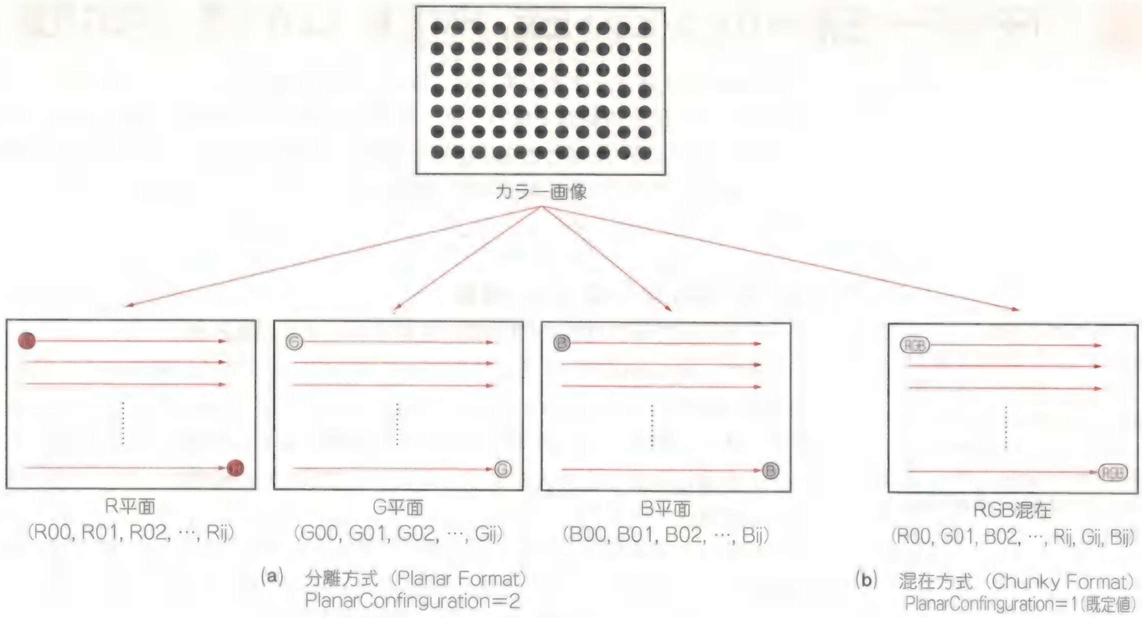
'0' ＝白のモノクロ画像
各ドットの画像データが '0' なら白，'1' なら黒を表すモノクロ画像．

'0' ＝黒のモノクロ画像
各ドットの画像データが '0' なら黒，'1' なら白を表すモノクロ画像．

RGB のプレーン
初期のパソコンの VRAM は R，G，B のプレーンが独立していて，それらを合成して色を表示していた．

Chunky Format
チャンク（Chunk）は，もともと肉やチーズなどの厚切りのかけらを意味している．この場合の Chunky Format というのは，そういうかけらをごちゃまぜにしたもの，という語感から名付けられたようだ．

〈図 2-8〉カラー・プレーン



しない方式が標準として用いられています．

プレーンを分離する方式を使いたいときは，PlanarConfiguration タグでパラメータ = 2 (Planar Format) を指定します．この PlanarConfiguration タグはベースライン TIFF にありますが，必須ではありません．

また，パレット・カラーの場合，4 ビットまたは 8 ビットのサンプル・データと実際の色を対応付けるカラー・パレットが必要です．ColorMap タグによってカラー・パレットを定義します．ColorMap タグのパラメータとして，実際の色を示す 16 個または 256 個の RGB データを書き並べることができます．

ピクセル数と画像の実サイズ

ディスプレイやプリンタは固有の解像度（ピクセル密度）が決まっています．画像の 1 ピクセルをディスプレイやプリンタの 1 ピクセルとしてそのまま表示すると，画像の実サイズはディスプレイやプリンタのピクセル密度によって変わってしまいます．また，**ピクセル数の多い高解像度の画像は大きく**，ピクセル数の少ない低解像度の画像は小さく表示されてしまいます．

しかし，ビット・マップ画像を表示する場合，画像の実サイズを自由に選びたい場合も少なくありません．そのためには，表示される画像の実サイズを指定するか，ピクセル密度を指定するかのどちらかが必要です．

TIFF では，ピクセル密度を指定するためのタグとして XResolution, YResolution, ResolutionUnit の二つが用意されています．ResolutionUnit はピクセル密度を計る単位長さを指定します．ResolutionUnit = 1 は inch, ResolutionUnit = 2 は cm で，ピクセル密度はそれぞれピクセル/inch，ピクセル/cm となります．また，とくに長さを指定しない（無単位）こともできます．

XResolution と YResolution では，単位長さの中に何個のピクセルが入るかを指定します．

ピクセル数の多い高解像度の画像は大きく
ディスプレイ表示の場合，実サイズの調整を行わずに，画像データの 1 ピクセルをディスプレイの 1 ピクセルに直接対応させてしまうのが普通だろう．

TIFFのデータ圧縮―パック・ビット圧縮，HM圧縮，LZW圧縮，JPEG圧縮

TIFFは非圧縮データとともにいくつかの圧縮方式もサポートしています．ベースラインTIFFの範囲ではパック・ビット圧縮とMH圧縮，拡張TIFFではそれに加えてLZW圧縮とJPEG圧縮も選択できます．ただし，いろいろな問題から，最近ではこれらの圧縮が実際に使われることは次第に少なくなってきています．

● バイト単位のパック・ビット圧縮

ラン・レングス方式
同じデータがN回繰り返されたら，それを1個のデータと数値N（繰り返し回数）に置き換える方式．この繰り返し回数のことをラン長（Run Length）と呼ぶ．圧縮方式としてはもっとも基本的なものの一つ．

バイト単位で**ラン・レングス方式**の圧縮を行います（**図2-9**）．

圧縮時，元の画像データをバイトごとに読んでいったとき，同じバイトが$n+1$回繰り返されたら，そのnバイトを数値$-n$と1バイト分のデータで置き換えます．また，同じバイトの繰り返しがない状態が$n+1$回続いたら，その$n+1$バイトの前に数値nを挿入します．ただし，数値$-n$の範囲は-1～-127，数値nの範囲は0～127とします．

伸長時，まず最初の1バイトを変数nに読み込みます．$0 \leqq n \leqq 127$ならそれに続く$n+1$バイトを画像データとします．また，$-127 \leqq n \leqq -1$ならその次の1バイトを$-n+1$回繰り返したものを画像データとします．さらに，次の1バイトを変数nに読み込んで同じことを繰り返します．

この圧縮/伸長は画像データの各行ごとに行います．行頭から1バイト単位で圧縮を行い，行末に1バイト未満のデータが残ったらそれは圧縮なしとします．そして，次の行の行頭からふたたび圧縮を行います．

対応できる画像の種類は広く，2値画像（1ビット/ピクセル）なら8ピクセル単位で，4ビット/ピクセルのグレー・スケール/パレット・カラーなら2ピクセル単位で，8ビット/ピクセルのグレー・スケール/パレット・カラーなら1ピクセル単位で圧縮を行います．

RGBカラーの場合，RGBのプレーンを分離して扱う方式（Planar Format）なら，各プレーンは8ビット/ピクセルとなりますから，1ピクセル単位で圧縮ができます．

RGBのプレーンを分離しない方式（Chunky Format）では，画像データはRGBRGBRGB…と並びます．同じバイトが繰り返されるのは，R成分とG成分とB成分が同じ値のときに限られますから，その確率はきわめて低くなります．圧縮アルゴリズムを実行することは可能ですが，圧縮の効果はまったく期待できないでしょう．

パック・ビット方式は圧縮/伸長の処理がきわめて簡単であり，Macintoshでは標準ツール・ボックスでサポートしています．しかし，圧縮効率はあまり高いとは言えず，とくに自然画像では圧縮の効果はあまり期待できません．

〈図2-9〉パック・ビット圧縮

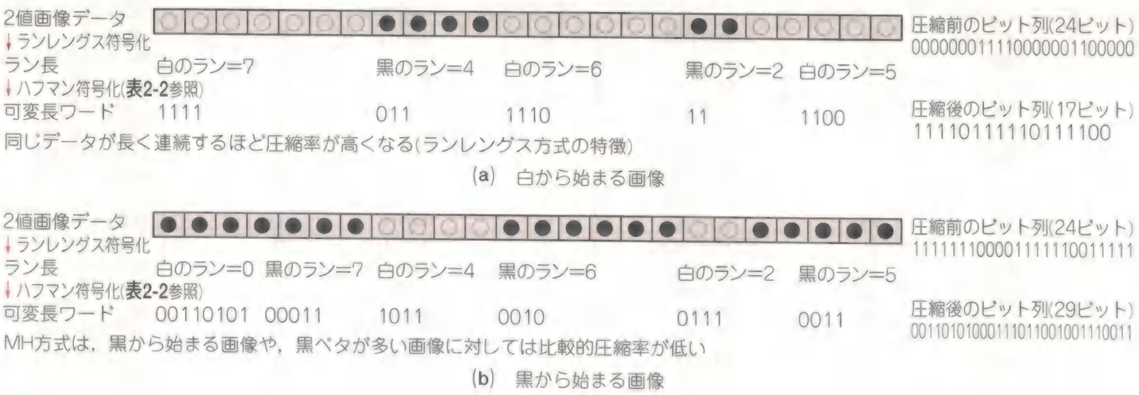
バイト列　A A A A A B C B C D E E E E F G H I J K L M M M

↓圧縮

数値&データ　−5 A　4 B C B C D　−4 E　6 F G H I J K L　−3 M

A～Mはそれぞれ異なる1バイトの値とする．
同じ値が長く連続するほど圧縮率が高くなる．この例では，24バイト→20バイトに圧縮されている．

〈図2-10〉MH圧縮



● MH修正ハフマン方式の圧縮

ラン・レングス方式とハフマン符号化を組み合わせたもので，**修正ハフマン**(Modified Huffman)**方式**と呼ばれています．対象は2値画像に限られますが，可逆的に高効率の圧縮ができます．ハフマン符号はエントロピ符号の代表的なものであり，JPEGでも使われています(**図2-10**)．

1行分の2値画像データを，白のラン長，黒のラン長，白のラン長，…，というように交互にラン長に変換します．画像が黒の部分から始まる場合でも，その前に長さ0の白のランを挿入することにより，かならず白のランから始まるようにします．

その後で，あらかじめ定められた変換テーブル(**表2-2**)を用いて，白と黒のラン長をそれぞれハフマン符号化します．任意長のランを符号化できるように，**ターミネーティング符号**(0～63のラン長に対応)と**メークアップ符号**(64，128，192，256，…，2496，2560のラン長に対応)を組み合わせる方式を採用しています．

ラン長が0～63のときは，直接1個のターミネーティング符号に符号化します．ラン長が63を超えると，ラン長が64～127ならメークアップ符号の64と0～63のターミネーティング符号の組み合わせ，ラン長が128～255ならメークアップ符号の128と0～63のターミネーティング符号の組み合わせ，…，ラン長が2560～2623ならメークアップ符号の2560と0～63のターミネーティング符号の組み合わせ，というようにメークアップ符号とターミネーティング符号を用いて符号化します．さらに，ラン長が2623を超えると，ラン長が2624～2687ならメークアップ符号の2560と64と0～63のターミネーティング符号の組み合わせ，…，というように複数のメークアップ符号とターミネーティング符号を用いて符号化します．これによって，どのような長さのランにも対応することができます．

このMH方式はもともとG3FAX(ITU-T T.4)で使われているものの流用で，スキャナ読み取りを前提にしています．変換テーブルは固定ですが，とくに白地に黒の文字や線画が描かれたデータを高効率に圧縮できるように選ばれています．

たとえば，黒のラン長2と3には2ビット符号の‘11’と‘10’，黒のラン長1と4には3ビット符号の‘010’と‘011’，黒のラン長5と6には4ビット符号の‘0011’と‘0010’というように，可変長のハフマン符号の中でとくに短いものが割

修正ハフマン方式

ハフマン(Huffman)方式も，圧縮方式としてはもっとも基本的なものの一つ．出現するデータを可変長の符号に置き換え，このとき出現頻度が高いデータに短い符号を，出現頻度が低いデータには長い符号を割り当てる．

ターミネーティング符号

ラン長を1個または複数の符号に置き換えるとき，最後はかならずこのターミネーティング符号で終わる．そのため，ターミネーティング(終止する)と名付けられた．

メークアップ符号

Make Upは作り上げる，構成するという意味．かならずターミネーティング符号と組み合わせて使う．

〈表 2-2〉 MH 圧縮の符号表

白のラン長	可変長ワード	白のラン長	可変長ワード
0	00110101	32	00011011
1	000111	33	00010010
2	0111	34	00010011
3	1000	35	00010100
4	1011	36	00010101
5	1100	37	00010110
6	1110	38	00010111
7	1111	39	00101000
8	10011	40	00101001
9	10100	41	00101010
10	00111	42	00101011
11	01000	43	00101100
12	001000	44	00101101
13	000011	45	00000100
14	110100	46	00000101
15	110101	47	00001010
16	101010	48	00001011
17	101011	49	01010010
18	0100111	50	01010011
19	0001100	51	01010100
20	0001000	52	01010101
21	0010111	53	00100100
22	0000011	54	00100101
23	0000100	55	01011000
24	0101000	56	01011001
25	0101011	57	01011010
26	0010011	58	01011011
27	0100100	59	01001010
28	0011000	60	01001011
29	00000010	61	00110010
30	00000011	62	00110011
31	00011010	63	00110100

黒のラン長	可変長ワード	黒のラン長	可変長ワード
0	0000110111	32	000001101010
1	010	33	000001101011
2	11	34	000011010010
3	10	35	000011010011
4	011	36	000011010100
5	0011	37	000011010101
6	0010	38	000011010110
7	00011	39	000011010111
8	000101	40	000001101100
9	000100	41	000001101101
10	0000100	42	000011011010
11	0000101	43	000011011011
12	0000111	44	000001010100
13	00000100	45	000001010101
14	00000111	46	000001010110
15	000011000	47	000001010111
16	0000010111	48	000001100100
17	0000011000	49	000001100101
18	0000001000	50	000001010010
19	00001100111	51	000001010011
20	00001101000	52	000000100100
21	00001101100	53	000000110111
22	00000110111	54	000000111000
23	00000101000	55	000000100111
24	00000010111	56	000000101000
25	00000011000	57	000001011000
26	000011001010	58	000001011001
27	000011001011	59	000000101011
28	000011001100	60	000000101100
29	000011001101	61	000001011010
30	000001101000	62	000001100110
31	000001101001	63	000001100111

(a) ターミネイティング符号

白のラン長	可変長ワード
64	11011
128	10010
192	010111
256	0110111
320	00110110
384	00110111
448	01100100
512	01100101
576	01101000
640	01100111
704	011001100
768	011001101
832	011010010
896	011010011
960	011010100
1024	011010101
1088	011010110
1152	011010111
1216	011011000
1280	011011001
1344	011011010
1408	011011011
1472	010011000
1536	010011001
1600	010011010
1664	011000
1728	010011011
EOL	000000000001

黒のラン長	可変長ワード
64	0000001111
128	000011001000
192	000011001001
256	000001011011
320	000000110011
384	000000110100
448	000000110101
512	0000001101100
576	0000001101101
640	0000001001010
704	0000001001011
768	0000001001100
832	0000001001101
896	0000001110010
960	0000001110011
1024	0000001110100
1088	0000001110101
1152	0000001110110
1216	0000001110111
1280	0000001010010
1344	0000001010011
1408	0000001010100
1472	0000001010101
1536	0000001011010
1600	0000001011011
1664	0000001100100
1728	0000001100101
EOL	00000000000

(b) メークアップ符号

〈図2-11〉LZW圧縮

(a) 入力データ列の例　IN[i]　A B A B C D A B C A B C D …　入力データ列はすべて1バイト

(b) 一致バッファ　Ω　可変長，初期値はφ(空)

(c) 入力バッファ　K　1バイト，初期値はφ(空)

(d) 出力データ列　OUT[i]　出力データ列は8～12ビット可変長

(e) 辞書の例

コード	データ
0x00	A(登録済み)
0x01	B(登録済み)
0x02	C(登録済み)
0x03	D(登録済み)
0x04	(登録済み)
…	(登録済み)
0xFF	(登録済み)
0x100	Clear
0x101	EOI
0x102	(未登録)
0x103	(未登録)
0x104	(未登録)
…	(未登録)
0xFFF	(未登録)

1バイトのデータはすべて登録済み

9バイトのデータの最初の二つ(0x100, 0x101)は予約語

新しいデータは0x102以降に登録する

0xFFFまで登録したら(辞書が満杯)，Clearを使って空にできる

(f) 圧縮の手順

① i=0
② K=IN[i]
③ Ω+Kは登録済み?（No→⑥）
④ Ω=Ω+K
⑤ i=i+1
②に戻る
⑥ Ω+Kを辞書に登録する
⑦ Ωのコードを出力データ列に書き出す
⑧ Ω=K
⑨ i=i+1
②に戻る

(g) 圧縮の例

②③ i=0，K=A，Ω+K=A(登録済み)
④ Ω=A
②③ i=1，K=B，Ω+K=AB(未登録)
⑥⑦ Ω+K=ABを0x102に登録，Ω=A(0x00)を出力
⑧ Ω=B
②③ i=2，K=A，Ω+K=BA(未登録)
⑥⑦ Ω+K=BAを0x103に登録，Ω=B(0x01)を出力
⑧ Ω=A
②③ i=3，K=B，Ω+K=AB(登録済み)
④ Ω=AB
②③ i=4，K=C，Ω+K=ABC(未登録)
⑥⑦ Ω+K=ABCを0x104に登録，Ω=AB(0x102)を出力
⑧ Ω=C
②③ i=5，K=D，Ω+K=CD(未登録)
⑥⑦ Ω+K=CDを0x105に登録，Ω=C(0x02)を出力
⑧ Ω=D
②③ i=6，K=A，Ω+K=DA(未登録)
⑥⑦ Ω+K=DAを0x106に登録，Ω=D(0x03)を出力
⑧ Ω=A
②③ i=7，K=B，Ω+K=AB(登録済み)
④ Ω=AB
②③ i=8，K=C，Ω+K=ABC(登録済み)
④ Ω=ABC
②③ i=9，K=A，Ω+K=ABCA(未登録)
⑥⑦ Ω+K=ABCAを0x107に登録，Ω=ABC(0x104)を出力
⑧ Ω=A
②③ i=10，K=B，Ω+K=AB(登録済み)
④ Ω=AB
②③ i=11，K=C，Ω+K=ABC(登録済み)
④ Ω=ABC
②③ i=12，K=D，Ω+K=ABCD(未登録)
⑥⑦ Ω+K=ABCDを0x108に登録，Ω=ABC(0x104)を出力
⑧ Ω=D

ここまでの出力データ列は，0x00 0x01 0x102 0x02 0x03 0x104 0x104

り当てられます．しかし，それ以外の黒のラン長に対しては，白のラン長より長い符号が割り当てられています．画像の黒ベタの部分は，白ベタより圧縮率が低くなります．

● さまざまな画像データの圧縮に使われるLZW圧縮

LZW (Lempel-Ziv & Welch) 方式は，さまざまな種類の画像データを可逆的かつ高効率に圧縮できる優れた符号化方式として知られています．1984年に発表されたもので，LZWという名称は3人の開発者の頭文字を並べたものです．GIFやTIFF, Postscript, PDFなど多くの規格に採用されています．

TIFFでは．Rev. 5.0からこのLZW圧縮をサポートしています．この頃はLZW方式は特許などの制約がなく自由に使えるものと考えられていましたが，Unisys社が特許を保有していることが後でわかりました．Rev. 4.0までのTIFFはカラー画像を効率よく圧縮する手段がなく，おもに非圧縮で利用されていました．LZW圧縮のサポートはユーザの広い支持を集めましたが，ライセンスの問

LZWという名称は3人の開発者の頭文字

もう少し詳しく言えば，Lempel氏とZiv氏が開発したLZ方式を，Welch氏が独自に改良したもの．LZ方式にはLZ77方式（1977年）とLZ78方式（1978年）の二つがあり，どちらもその後のデータ圧縮技術の基礎となった有名な方式．このLZ方式には，さらにいろいろな改良が加えられて，LZ～方式という新しい方式が次々に作られていった．LZWもその一つ．

題もあり，最近はふたたび非圧縮での利用が多くなっているようです．

8ビットのキャラクタの列
LZ方式はもともとテキスト情報の暗号化や圧縮の研究から生まれてきたもののようだ．入力データとしてはキャラクタ（たとえばASCIIコード）が想定されており，パターンの比較を行うのに辞書という表現を使っている．もちろん，テキスト・データだけでなく，画像データやランダムなバイナリ・データについても同様に利用できる．

TIFFのLZW圧縮では，**8ビットのキャラクタの列**として与えられるデータを，8～12ビットの可変長符号に符号化します．データを順次調べて出現したパターンを辞書に記録するとともに，データを符号に置き換えていきます．データを表示するときは，符号化されたデータを順次調べて出現した符号を辞書に記録するとともに，符号をデータ・パターンに置き換えます（**図2-11**）．LZW圧縮については第1-4章のGIFを参照してください．

元の画像データが1ビット/ピクセル，4ビット/ピクセルなどの場合は，8ビット分ずつまとめて扱います．また，元の画像データが16ビット/ピクセル（YCbCr），24ビット/ピクセル（RGB），32ビット/ピクセル（CMYK）などの場合は，8ビットずつに分けて扱います．

また，TIFFのLZW圧縮では，もともとのLZW符号に含まれていないクリア符号とEOI（End Of Information）符号の二つが追加されています．クリア符号は辞書が満杯になったときに辞書をクリアして新しくするための符号で，GIFで導入されたものです．

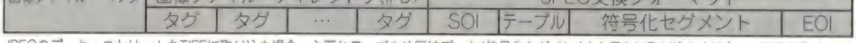
TIFFでは，LZW圧縮の前処理として差分予測処理を行うことができます．1ラインの中で，先行（左側）の1ピクセルの値を予測値とし，予測値との差をとってキャラクタ列を生成します．同じ値が連続すれば差分値は0の連続となり，高い圧縮率が得られます．

差分予測処理によって圧縮率が向上するかどうかは，実際の画像のパターンによって異なります．TIFFファイルが差分予測を行っているかどうかの情報は，Predictorタグに書き込まれます．Predictor＝1は差分予測なし，Predictor＝2は差分予測ありです．

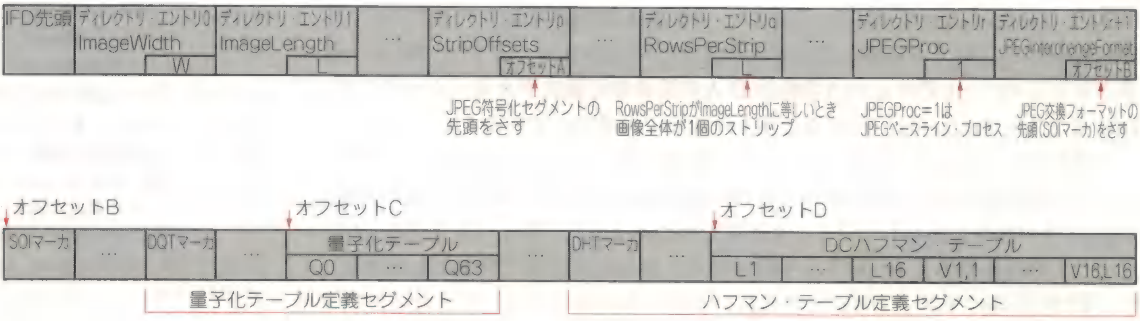
TIFFではグレー・スケールとRGBカラー，YCbCrカラー，CMYKカラーに限定したJPEG圧縮

TIFFでは，Rev. 6.0からはJPEG圧縮もサポートするようになりました．JPEGのうち，ベースライン・プロセス（基本DCT利用型プロセス）と可逆プロセスが利用可能です．ただし，可逆プロセスは圧縮効率が低いため実際にはあま

〈図2-12〉JPEGデータ・ストリームの取り込み



JPEGのデータ・ストリームをTIFFに取り込む場合，必要なテーブルや圧縮データ（符号化セグメント）を個々に取り込む方法と，JPEG交換フォーマット全体をまとめて取り込む方法が考えられる．JPEG交換フォーマットをまとめて取り込んだ場合，通常のJPEGファイル処理と同様にSOIから順次処理していく方法と，必要なテーブルや符号化セグメントを個別に処理する方法が考えられる．TIFFのタグの中には，テーブルや符号化セグメントのオフセットを個別に指定できる機能をもつものがある．



り利用されていません．

ベースライン・プロセスは非可逆であり，原画像にくらべて多かれ少なかれ画質の低下は避けられません．そのかわり，自然画像データをきわめて高効率に圧縮できる方式として知られています．一方，2値画像やパレット・カラーに対しては圧縮率が低いだけでなく，画質の劣化が目立ちます．そのため，TIFFではJPEG圧縮の対象をグレー・スケール（PhotometoricInterpretation＝1かつBitsPerSample＝4または8）とRGBカラー（PhotometoricInterpretation＝2）/YCbCrカラー（PhotometoricInterpretation＝6）/CMYKカラー（PhotometoricInterpretation＝5）に限定しています．

JPEGの規格ではファイル・フォーマットの規定が十分でなく，画像再生に必要な情報の一部が不足する場合がありました．TIFFとJPEGを組み合わせれば，不足する情報はTIFFのタグで補うことができます．そのため，**TIFFによるJPEGのサポート**は多くのユーザに歓迎され，画像フォーマットの主流になるのではないかと期待された時期もあります．しかし，TIFFのJPEG拡張にもいくつかの問題点があることがわかり，またJPEGの規格の不十分な部分を補うJFIF（JPEG File Interchange Format）が普及したこともあって，TIFFのJPEG圧縮を利用することは次第に少なくなっています．

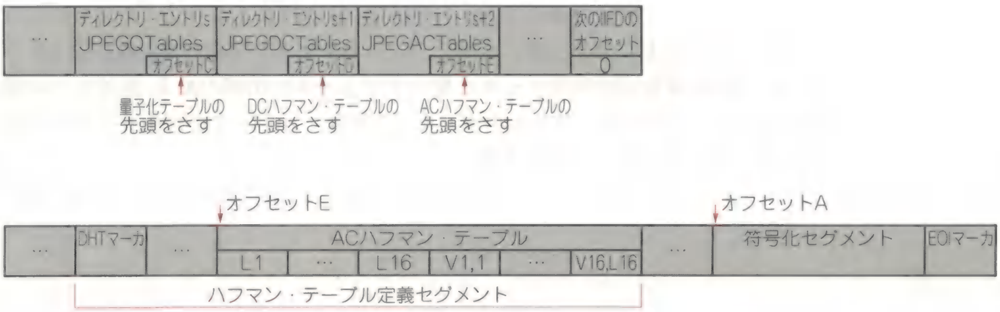
JPEGでは，圧縮（符号化）した画像データにマーカやパラメータを付加して，SOI（画像開始）マーカで始まりEOI（画像終了）マーカで終わる1列のデータ・ストリーム（交換フォーマット）を構成します．元の画像データに関する情報や，圧縮/伸長のために必要な情報（量子化テーブル，DC/ACハフマン・テーブルなど）はこのデータ・ストリームに含まれます．

TIFFでは，このJPEG交換フォーマット全体をTIFFファイルの中に取り込むことができます．それに，TIFFで必要とする情報をタグとして付加します（**図2-12**）．とくに，JPEGの仕様には画像の実サイズに関する情報など欠落しているものがあり，それはTIFFのタグで補うことができます．

一方，JPEG交換フォーマット全体を取り込む代わりに，圧縮（符号化）した画像データだけをTIFFファイルに取り込むこともできます．この場合も，TIFFで必要とする元の画像データに関する情報や，圧縮/伸長のために必要な

TIFFによるJPEGのサポート

TIFFの規格はJPEGよりずっと早くから普及していたし，すでにパック・ビット，MH，LZWなど複数の圧縮方式をサポートしていた．そのため，JPEGの最終規格案が発表された1990年頃から，この圧縮方式をTIFFの中でも使いたいというユーザの期待が大きく高まっていた．JPEG規格がITU-Tで正式に承認されたのは1992年9月だが，JPEGをサポートするTIFF Rev. 6.0はそれよりも少し早い1992年6月にすでに発表されている．



〈表 2-3〉JPEG のためのタグ

名称	コード	パラメータ型	ベースライン・プロセス	可逆プロセス
JPEGProc	0x0200	SHORT	JPEGProc = 1	JPEGProc = 14
JPEGInterchangeFormat	0x0201	LONG	○	○
JPEGInterchangeFormatLength	0x0202	LONG	○	○
JPEGRestartInterval	0x0203	SHORT	○	○
JPEGLosslessPredictors	0x0205	SHORT	×	○
JPEGPointTransforms	0x0206	SHORT	×	○
JPEGQTables	0x0207	LONG	○	×
JPEGDCTables	0x0208	LONG	○	○
JPEGACTables	0x0209	LONG	○	×

情報（量子化テーブル，DC/AC ハフマン・テーブルなど）は，タグとして付加しなければなりません．

JPEG 圧縮を利用する場合，Compression タグで 6（JPEG 圧縮）を選ぶとともに，**表 2-3** のようなタグを付加することが必要です．

JPEGProc タグは JPEG のプロセスを選択するもので，JPEGProc = 1 がベースライン・プロセス，JPEGProc = 14 が可逆プロセスです．JPEGInterchangeFormat タグを用いて JPEG 交換フォーマットの先頭（SOI マーカ）のオフセットを指定し，JPEGInterchangeFormatLength タグを用いて JPEG 交換フォーマットのバイト数を指定します．

TIFF ファイルが JPEG 交換フォーマット全体を含まない場合は，JPEGInterchangeFormat = 0 とし，JPEGInterchangeFormatLength は不要です．

JPEG でリスタート・マーカを用いている場合は，JPEGRestartInterval タグでリスタートのインターバルを指定します．

ファイルが JPEG 交換フォーマット全体を含む場合も，そうでない場合も，JPEGQTables タグを用いて量子化テーブルのオフセットを，JPEGDCTables タグを用いて DC ハフマン・テーブルのオフセットを，JPEGACTables タグを用いて AC ハフマン・テーブルのオフセットを，それぞれ指定します．JPEG ストリームを伸長（復号化）するためには，これらのテーブルが必要です．

TIFF では画像をストリップまたはタイルに分割して扱いますが，JPEG では画像全体のデータを一括して圧縮します．そのため，通常は TIFF でも画像全体を **1 本のストリップもしくは 1 個のタイル**として扱います．そして，StripOffsets（もしくは TileOffsets）タグを用いて圧縮データの先頭オフセットを指定します．

1 本のストリップもしくは 1 個のタイル

ベースライン TIFF の範囲では，画像は横幅いっぱいの帯（ストリップ）を組み合わせたものとして扱う．拡張 TIFF では，画像を縦横に分割した適当なサイズのタイルを組み合わせたものとして扱うことができる．

ただし，JPEG 圧縮でリスタート・マーカを用いている場合は，JPEG でも画像をリスタートごとに分割して扱うことができます．この場合，StripOffsets（もしくは TileOffsets）タグを用いて，圧縮データの先頭および各リスタート後の先頭オフセットを指定することができます．

このように，TIFF ではオフセットを指定して JPEG ストリームを部分的に取り出すことが容易にできます．ただし，このオフセット指定には危険な面もあります．JPEG 非対応の画像ツールを使ってファイルに情報の挿入/削除などの操作を行うと，JPEG ストリームのオフセットが狂い，データの再生が正常にできなくなってしまうことがあります．

第1-3章　Windowsで標準として使われている画像フォーマット

DIB

宮崎　仁

〈名称〉
Device Independent Bitmap
〈発行者〉
Microsoft Corp.
〈情報入手先〉
http://www. microsoft. com/（Microsoft社）
〈参考資料〉
マイクロソフトウィンドウズソフトウェア開発キットプログラマーズリファレンスVol. 3メッセージ，構造体，マクロ（1993年，翔泳社）
マイクロソフトウィンドウズソフトウェア開発キットプログラマーズリファレンスVol. 4リソース（1993年，翔泳社）

BMPファイルはWindowsプログラミングの標準画像フォーマット

DIB（Device Independent Bitmap）は，Windowsで標準として使われている画像フォーマットです．ファイルの拡張子としては．BMPが用いられることが多く，一般には**BMPファイル**とも呼ばれています．また，DIBはファイル・フォーマットだけでなく，Windowsアプリケーションがメモリ上でビット・マップを扱うときの標準形式もDIBと呼ばれます．

BMPファイル
拡張子としてDIBが用いられることもある．また，RLE圧縮された場合には拡張子としてRLEを用いることもある．なお，Windows形式だけでなく，OS/2形式のDIBも同様に拡張子BMP, DIB, RLEなどを用いることがある．さらに，DIBでない（デバイス独立でない）ビットマップ・ファイルにも拡張子BMPが用いられることがある．拡張子だけでは，ファイルの種類は正確に判定できない．

2色（モノクロ）および16色（4ビット・カラー），256色（8ビット・カラー），1677万色（24ビット・カラー）などのビット・マップ画像を扱うことができます．256色以下のDIBフォーマットは付加情報としてカラー・パレットをもっており，表示デバイスの種類に依存せずに同じ色で表示できます．デバイス独立（Device Independent）と呼ばれるのはこのためです．

主として非圧縮で利用されますが，16色と256色ではRLE（Run Length Encoding）方式の圧縮もサポートしています．

Windowsプログラミングでは DIBを標準的に使用するので，DIBを扱うためのグラフィック関数ライブラリが提供されています．DIBの仕様についてはMicrosoft社が提供するWindows開発環境のマニュアルに解説されています．書籍として単独で入手できるものとしては，Windows 3.1用のSDK（Software Development Kit）の日本語版マニュアルが翔泳社から発売されていましたが，現在はおそらく絶版でしょう．

GUIのために作られたDIB画像フォーマットの成り立ち

Microsoft社では，1983年にMS-DOS上でグラフィック・ユーザ・インターフェース（GUI）を提供するMicrosoft Windowsの構想を発表し，1985年にWindows 1.0，1987年にWindows 2.0を発売しました．当時のWindowsで使わ

れていた標準のビット・マップ形式は，特定の表示デバイス（ビデオ・カード）の方式に依存するもので，同じビット・マップ・データでも表示デバイスが異なると，まったく違った表示になってしまうものでした．この旧式のビット・マップ形式は，その後 DDB（Device Dependent Bitmap）と呼ばれるようになりました．

一方，Microsoft 社では IBM PS/2 用の OS として 1987 年に OS/2 を IBM 社と共同開発しました．OS/2 そのものはテキスト・ベースの OS でしたが，1988 年に発売された OS/2 1.1 にはプレゼンテーション・マネージャと呼ばれる新しいグラフィック・ユーザ・インターフェース（GUI）が付属していました．このプレゼンテーション・マネージャにおいて，表示デバイスに依存しないデバイス独立ビット・マップ（DIB）が導入されました．

1990 年に発売された Windows 3.0 では，この OS/2 の DIB を拡張したものが標準として取り入れられ，それ以降の Windows 3.1，Windows 95，Windows 98 にも引き継がれています．なお，Windows 標準の DIB は OS/2 の DIB と互換性はありません．そのかわり，Windows では標準の DIB のほかに OS/2 互換形式もサポートしています．

DIB はもともと異機種間でのファイル交換を目的としたものではなく，非 Windows 環境での利用は考慮されていません．Windows の普及とともに DIB も広く普及したため，最近では非 Windows アプリケーションでも DIB をサポートする例が多いようです．ただし，WORD（2 バイト）/DWORD（4 バイト）データのバイトの並び順が機種依存（リトル・エンディアン）であること，RGB を独立プレーンとして扱えないこと，ビット・マップが通常とは逆に画像の下から上に並ぶことなど，汎用画像フォーマットとしては扱いにくい仕様であると言われています．

OS/2

IBM PS/2 とともに初めて発表されたときは，MS-DOS の拡張版という感じだったが，その後 GUI を取り入れて改良が進められ，一時は Windows の対抗馬としてがんばっていた（OS/2 Warp）．

Windows 3.0

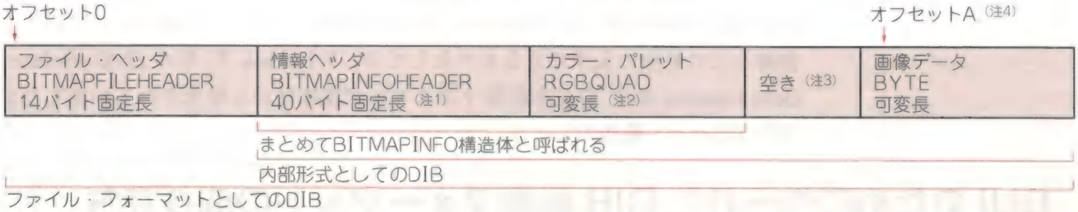
Microsoft 社では多額の資金と人材，時間を投入して Windows を開発してきたが，1.x から 2.x まではあまり使いものにならず，会社の将来性にまで疑問が投げかけられるほどだった．Windows 3.0 はやっとまともに使えるものとなり，市場でも好評だった．そこから Microsoft 社の反撃が始まり，やがて Windows が市場を制覇するまでになった．

標準構成は BITMAPINFOHEADER，RGBQUAD，BYTE の三つの部分

Windows 標準の DIB は，画像に関する情報が記述された BITMAP INFOHEADER，カラー・パレットを定義する RGBQUAD，画像データ本体である BYTE の三つの部分からなります．これをファイル化したもの（BMP ファイル）には，最初にファイルに関する情報が記述された BITMAPFILEHEADER が追加されます（**図 3-1**）．

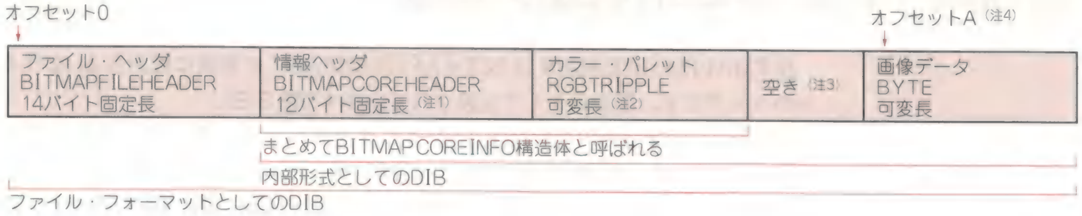
OS/2 互換形式の DIB では，BITMAPINFOHEADER のかわりに BITMAPCOREHEADER，RGBQUAD のかわりに RGBTRIPPLE が定義されて

〈図 3-1〉DIB の基本構成



（注1）BITMAPINFOHEADERの先頭4バイトに，ヘッダ長0×00000024(10進で40)が書き込まれている．
（注2）24ビット・カラーでは使わない．
（注3）BITMAPFILEHEADERからRGBQUADまでは連続．画像データはその後の任意のワード境界(偶数アドレス)から開始できる．
（注4）画像データの先頭オフセットは，BITMAPFILEHEADERの中で指定する．

〈図 3-2〉OS/2 形式の DIB



(注1) BITMAPCOREHEADERの先頭4バイトに，ヘッダ長0×0000000C(10進で12)が書き込まれている．
(注2) 24ビット・カラーでは使わない．
(注3) BITMAPFILEHEADERからRGBTRIPPLEまでは連続．画像データはその後の任意のワード境界(偶数アドレス)から開始できる．
(注4) 画像データの先頭オフセットは，BITMAPFILEHEADERの中で指定する．

います (**図 3-2**)．

BITMAPFILEHEADER, BITMAPINFOHEADER, RGBQUAD は連続した領域に置かれます．これらのフォーマットは，それぞれ構造体として定義されています．なお，24 ビット・カラーではパレット情報は不要なので，RGBQUAD の部分はありません．

画像データ本体は BYTE の配列です．RGBQUAD の後の任意のオフセット（ファイル先頭を 0 とする相対アドレス）から開始することができます．ただし，画像データの先頭アドレスはワード境界（偶数アドレス）とします．また，非圧縮の場合，1 行分のデータが DWORD（4 バイト）の整数倍になるように，必要に応じて行末に 0 を付加します．

構造体
もともと DIB は Windows プログラミングのためのフォーマットなので，厳密な定義はライブラリ関数のヘッダなどに含まれている．

BITMAPFILEHEADER 構造体の構成

BMP ファイルの先頭に置かれる 14 バイトのヘッダです．構造体として定義されています (**表 3-1**)．

bfType はビット・マップ・ファイルであることを表すもので，常に文字 “BM” (0 x42, 0 x4D) が入ります．

bfSize はファイル・サイズ（合計のバイト数）を DWORD（4 バイト）で指定します．

bfReserved1, bfReserved2 はともに予約領域で，常に 0 とします．

bfOffBits は画像データ先頭のオフセットを DWORD（4 バイト）で指定します．

〈表 3-1〉BITMAPFILEHEADER の構成

フィールド	型	説明
bfType	WORD	DIB ファイルの識別子，常に ‘B’，‘M’ の 2 文字
bfSize	DWORD	ファイル全体のサイズ
bfReserved1	WORD	予約，常に 0
bfReserved2	WORD	予約，常に 0
bfOffbits	DWORD	画像データ (BYTE) の先頭オフセット

このファイル・ヘッダは Windows 形式，OS/2 形式とも共通．

BITMAPINFOHEADER 構造体の構成

BITMAPINFOHEADER は BITMAPFILEHEADE の直後に置かれる 40 バイトのヘッダです．構造体として定義されています（**表 3-2**）．

biSize は BITMAPINFOHEADER のサイズ（バイト数）で，これまでの Windows の DIB では常に 40（0 x00000028）です．

なお，OS/2 形式の DIB では，BITMAPFILEHEADE の直後に 12 バイトの BITMAPCOREHEADER が置かれます．そして，その 1 バイト目は biSize = 12（0 x0000000C）です．したがって，この biSize を参照すれば Windows 形式の DIB と OS/2 形式の DIB を見分けることができます．

biWidth は画像の幅，biHeight は画像の高さで，ともにピクセル数で指定します．

biPlanes は**画像のカラー・プレーン数で，常に 1 が入ります**．24 ビット・カラーの場合，1 ピクセルを連続した RGB の 3 バイトとして扱います．R，G，B 独立の 3 プレーンとしては扱いません．

画像のカラー・プレーン数で，常に 1 が入ります

このように，カラー・プレーン数を示すパラメータが用意されているので，Microsoft 社が必要を認めればいつでも拡張は可能だろう．

〈表 3-2〉BITMAPINFOHEADER の構成

フィールド	型	説明
biSize	DWORD	BITMAPINFOHEADER のサイズ．常に 0x00000028
biWidth	DWORD	ビット・マップの幅（ピクセル数）
biHeight	DWORD	ビット・マップの高さ（ピクセル数）
biPlanes	WORD	表示デバイスのカラー・プレーン数，常に 1
biBitCount	WORD	ピクセルあたりのビット数
biCompression	DWORD	圧縮形式
biSizeImage	DWORD	圧縮後の画像データ・サイズ
biXPelsPerMeter	DWORD	表示デバイスの水平解像度，必要なければ 0
biYPelsPerMeter	DWORD	表示デバイスの垂直解像度，必要なければ 0
biClrUsed	DWORD	このビット・マップが使用している色数．0 なら全色を使用
biClrImportant	DWORD	このビット・マップの表示にとくに重要な色数．0 なら全色が重要

(a)　ヘッダの構造

値	説明
0	Windows98 以降．JPEG 圧縮のための予約
1	1 ビット/ピクセル（モノクロ）．2 色（0 ＝背景色，1 ＝前景色）のカラー・パレットが必要
4	4 ビット/ピクセル．最大 16 色のカラー・パレットが必要
8	8 ビット/ピクセル．最大 256 色のカラー・パレットが必要
16	Windows95 以降．16 ビット/ピクセル（ハイカラー）．カラーパレットは使用しない（注）
24	24 ビット/ピクセル（フル・カラー）．カラー・パレットは使用しない（注）．1 ピクセルを 3 バイト（RGB 各 1 バイト）で表す
32	Windows95 以降．1 ピクセルを 4 バイト（RGB 各 1 バイトと NULL）で表す．表示色数は 24 ビット/ピクセルと同じ

（注）16 ビット/ピクセル以上では，画像の表示のためのカラーパレットは必要ない．
ただし，システムのカラー・パレットを最適化するためのパレット情報を付加できる．
このときは，biClrUsed フィールドにパレット・サイズを記述する．

(b)　ピクセルあたりのビット数

フィールド	値	説明
BI_RGB	0	圧縮なし．パレット・カラーでも圧縮しないときは BI_RGB を指定
BI_RLE8	1	8 ビット/ピクセルの RLE 圧縮
BI_RLE4	2	4 ビット/ピクセルの RLE 圧縮
BI_BITFIELDS	3	Windows95 以降．16/32 ビットのみ．圧縮なし．カラー・マスク使用
BI_JPEG	-	Windows98 以降の予約

(c)　圧縮形式

biBitCount は 1 ピクセルのビット数です．2 色は 1，16 色は 4，256 色は 8，RGB フル･カラーは 24 が入ります．なお，1 ビット/ピクセルというのは一般的には白黒画像ですが，DIB の場合はカラー･パレットで定義した任意の 2 色を用います．カラー･パレットで白と黒の 2 色を定義すれば，白黒画像とすることができます．

biCompression は圧縮形式を示すものです．BI_RGB（圧縮なし），BI_RLE8（8 ビット/ピクセルのラン･レングス圧縮），BI_RLE4（4 ビット/ピクセルのラン･レングス圧縮）の三つの圧縮形式が定義されており，biCompression＝0 は BI_RGB，biCompression＝1 は BI_RLE8，biCompression＝2 は BI_RLE4 を表します．2 値画像と RGB フルカラーではこの値は常に 0（圧縮なし）です．16 色では 0（圧縮なし）または 2（BI_RLE4），256 色では 0（圧縮なし）または 1（BI_RLE8）です．

biSizeImage は画像データのサイズです．圧縮を行った場合（biCompression＝1 または 2）のみ，圧縮後の画像データのバイト数をここに書き込みます．圧縮なし（biCompression＝0）のときはこの値を使用する必要はありません．

biXPelsPerMeter は表示デバイスの水平解像度，biYPelsPerMeter は表示デバイスの垂直解像度を **1 m あたりのピクセル数**で示したものです．画像ファイルを作成するときにアプリケーションが想定している表示デバイスの解像度をここに記述できます．画像を表示するアプリケーションはこの値を参照して，実際の表示デバイスの解像度がこれと異なるときには画像のビット･マップを補正して表示することができます．ただし，実際にはこれらの値はあまり使われておらず，0 が書き込まれていることが多いようです．

biClrUsed，biClrImportant はともにカラー･パレットに関する補助的な情報です．**表示色数の少ないデバイスを使って減色表示を行うとき**に，元の画像を損なわずに減色できる範囲を示します．ただし，実際にはこれらの値は使用されず，単に 0 が書き込まれていることが多いようです．

biClrUsed は，カラー・パレットに定義された色の中で，画像の中で実際に使用されている色数を示します．ただし，biClrUsed ＝ 0 は，16 色または 256 色すべてが使われていることを示します．また，画像が 24 ビット・カラーの場合は，表示デバイスでパレットに減色して表示するときに用意すべきテーブル・サイズを示します．

biClrImportant は，カラー･パレットに定義された色の中で，この画像を表示するために重要である色数を示します．RGBQUAD で定義されるカラー･パレットは，色の**重要度の順**に並べるように規定されています．したがって，biClrImportant＝N のとき，カラー･パレットの先頭から N 個の色数だけを表示すれば，減色による画質の低下を最小限にすることができます．

1m あたりのピクセル数
画像解像度をメートルあたりで定義するのは珍しいが，1 m は約 40 インチだから，600 dpi（dot per inch）なら約 24000 ドット/m となる．

表示色数の少ないデバイスを使って減色表示を行うとき
DIB が作られた頃はまだ VGA の時代で，パソコンのディスプレイは 16 色表示があたりまえだった．256 色の画像データを 16 色でどうやって自然に見せるか，という苦労も多かったのだろう．

重要度の順
別に一般的な基準があるわけではなく，画像の作成者が重要と思う順に並べなさい，ということ．しかし，実際に重要度の順を考えてカラー・パレットを並べ換える，などという手間をかける人はほとんどいないように思う．

RGBQUAD 構造体の構成

256 色以下（2 値画像およびパレット･カラー）の BMP ファイルでは，14 バイトの BITMAPFILEHEADER，40 バイトの BITMAPFILEHEADER に続いて，表示色を定義するカラー･パレットが置かれます．カラー･パレットは RGBQUAD 構造体（**表 3-3**）の配列として定義され，配列の先頭がパレット番号 0 に，配列の i 番目はパレット番号 $i-1$ に対応します．前述のように，パレット番号は原則として重要度の順に並べます．

とくに，2 値画像の場合はデータ＝0 はパレット番号 0，データ＝1 はパレッ

〈表 3-3〉カラー・パレット

フィールド	型	説明
rgbBlue	BYTE	B 成分の強さを示す
rgbGreen	BYTE	G 成分の強さを示す
rgbRed	BYTE	R 成分の強さを示す
rgbReserved	BYTE	使用しない．常に 0 (NULL)

(a) RGBQUAD

エントリ	データ
0	RGBQUAD[0]
1	RGBQUAD[1]
2	RGBQUAD[2]
3	RGBQUAD[3]
…	…
$N-1$	RGBQUAD[N − 1]
N	RGBQUAD[N]

$N = 1$ (1 ビット/ピクセル)
$1 \leqq N \leqq 15$ (4 ビット/ピクセル)
$1 \leqq N \leqq 255$ (8 ビット/ピクセル)

DIB のカラー・パレットは，RGBQUAD 構造体 (4 個の BYTE からなる) の配列．
ピクセル・データ i に RGBQUAD[i] が対応し，そのピクセルの表示色を与える．
その画像で使われているピクセル・データに対応する分だけあればよい．

(b) DIB のカラー・パレット

エントリ	パレット・データ			
0	B	G	R	NULL
1	B	G	R	NULL
2	B	G	R	NULL
3	B	G	R	NULL
…				
$N-1$	B	G	R	NULL
N	B	G	R	NULL

(c) パレット・データ

ト番号 1 に対応します．白黒表示だけでなく，任意の 2 色表示 (背景色 + 前景色) にできます．

画像を正しく表示するためには，カラー・パレットは画像データに含まれるパレット番号をすべて網羅していることが必要です．逆に，画像データで使われないパレット番号をカラー・パレットで定義していてもかまいません．2 値画像の DIB に 256 色のカラー・パレットを定義しても差し支えありません．

RGBQUAD 構造体は，パレット番号で与えられる色がもつ RGB 情報を定義するもので，第 1 バイトは rgbBlue，第 2 バイトは rgbGreen，第 3 バイトは rgbRed というように実際には BGR の順に並んでいます．また，**第 4 バイトは予約で常に 0** が入ります．

実際の表示デバイスはそれぞれ異なるカラー・パレットを参照して表示を行っていますが，BMP ファイルを表示するときはファイル内のカラー・パレットを参照すれば常に正しい表示ができます．

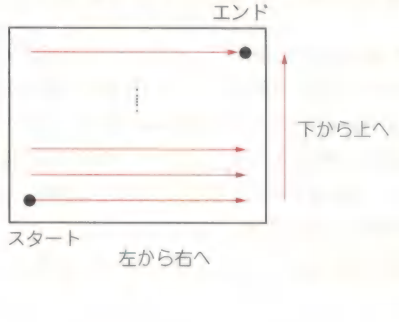
第 4 バイトは予約で常に 0
一般に，RGB データは 8 ビット × 3 = 24 ビットで表現できるが，コンピュータのデータ処理としては 32 ビット (4 バイト) 単位になっている方がずっとやりやすいので，余分な 1 バイトを付け加える方法もよく用いられる．

画像データの配列

画像データの部分はバイトの配列となります．ただし，表示色数によってその内容は異なります．また，DIB の画像データの並び順は，**一般的な画像データ**の並び順と違っています (図 3-3)．

一般的な画像データ
左上から始まってトップダウンの順にスキャンしていくのは，テレビの方式から引き継がれているようだ．そして，そのおおもとは西欧の言語の書き方 (左から右へ，上から下へ) からきていると思われる．

〈図 3-3〉画像データの並び順



一般的な画像データでは，画像の行は上から下（top down）の順で並べられ，1行の中では左から右にピクセルが並びます．Windows や OS/2 の DIB では，1行の中で左から右にピクセルが並ぶのは一般と同じですが，画像の行は下から上（bottom up）の順で並べられます．そのため，一般的な画像データは画面の左上から始まって右下で終わるのに対して，DIB の画像データは画面の左下から始まって右上で終わります．

2色画像（biBitCount＝1）の場合，1ピクセルを1ビットで表し，8ピクセル分の横並びのデータを1バイトで表します．このとき，MSB がもっとも左側の1ピクセルを表します．

16色画像（biBitCount＝4）の場合，1ピクセルを4ビットで表し，2ピクセル分の横並びのデータを1バイトで表します．MSB 側の4ビットが左側の1ピクセルを表します．

256色画像（biBitCount＝8）の場合，1ピクセルを1バイトで表します．

フルカラー画像（biBitCount＝24）の場合，1ピクセルを RGB の3バイトで表します．

画像の圧縮

DIB ではパレット・カラー（16/256色）の圧縮はできますが，よりファイル容量の大きいフルカラー（1677万色）の圧縮はサポートしていません．一般に，BMP ファイルは非圧縮なので大容量になると言われるのはそのためです．

16色表示（4ビット/ピクセル）と256色表示（8ビット/ピクセル）で使われる圧縮方式は，基本的には同じものですが，ビット数が違うため圧縮方式にも若干の違いがあります．

トランジスタ技術 SPECIAL　エレクトロニクスの基礎と実用技術を凝縮したフィールド・ワーク・マガジン
好評発売中!
No.63
B5判　176頁
定価 1,840円（税込）

特集　パソコン周辺インターフェースのすべて
PC を使いこなすためのハードウェア規格リファレンス

現在のパソコンは，すべてなんらかの規格によって造られています．いままでは，やったもの勝ち，あるいは市場で残ったものが生き残るというところがありましたが，これからは勝ち組といわれるマイクロソフト社とインテル社が主導的に規格化を進めていく傾向が強くなっています．標準化を進める国や国際機関もありますが，あまりの技術の進歩の速さに規格化が追いついていないのが現状です．
本書は，パソコンとバス，オーディオ／ビジュアル／マルチメディアの分野について，個々の規格の成り立ちからその内容，実現方法をふくめて，リファレンスとして役立つように解説を加えました．

CQ出版社　〒170-8461　東京都豊島区巣鴨 1-14-2 販売部　☎(03) 5395-2141　振替 00100-7-10665

● 8ビット/ピクセルの画像データの圧縮 BI_RLE8

8ビット/ピクセルの画像データに対して，1ピクセル単位でラン・レングス方式の圧縮を行います．データの圧縮を行うエンコード・モードと，**圧縮しない絶対モード**の2種類があり，1個のビット・マップの中で両者を混在させることができます(**図3-4**).

エンコード・モードは2バイトの組からなり，1バイト目がピクセル数N (N= 1～255)，2バイト目がデータを示します．画像データに同じ値がNピクセル分連続したとき（ラン長=N），Nバイトの元データを2バイトに圧縮できます．同じ値が長く連続するほど，圧縮効率は高くなります．

また，エンコード・モードにはラン長のほかに3種類のエスケープが定義されています．エスケープの1バイト目は常に0で，2バイト目の値はK =0，1，2のいずれかです．このKの値によってエスケープの意味が決まります．

K =0は1行の終端を示します．K =1はビット・マップ全体の終端を示します．K =2はピクセルが連続していないときの偏位を示すのに使われます．K =2に続く2バイト（ともに符号なし整数）によって次に表示されるピクセルの水平オフセット，垂直オフセットを示します．このK =2のエスケープは，**透過画像**を表現するために定義されたものと思われますが，マニュアルには明確な記述がなく実際にも使用されていないようです．

絶対モードはデータを圧縮せず，元のデータ列をそのまま使います．ただし，エンコード・モードと区別するために，Mピクセル分のデータ列の前に2バイトを付加します．1バイト目は常に0，2バイト目はピクセル数M (M =3～255)を示します．これによって，3バイト目から3+ M −1バイト目までは元データのままであることがわかります．さらに，Mが奇数の時は，末尾に1バイトの0を付加してデータ列のバイト数が偶数になるようにします．これによって，圧縮

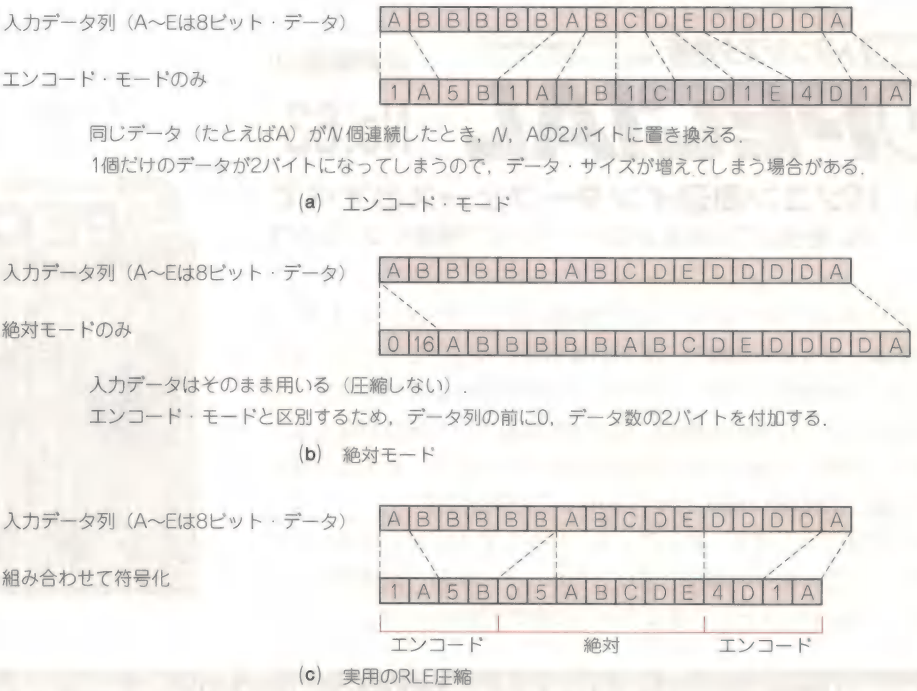
圧縮しない絶対モード

ラン・レングス圧縮は，同じデータが何個も続けばより圧縮率が高くなるが，1個ずつ異なるデータが続けば圧縮でないだけでなく，かえってデータ量が増えてしまう．そのため，同じデータが続いたときだけ圧縮し，そうでなければもとのデータをそのまま残す，という処理が必要．

透過画像

GIFなどでは，パレット・カラーのうち特定のパレット番号を透過色に指定し，画像データの中にその番号のドットがあったら表示せずに次のドットに進む，という方法で透過画像を表現している．DIBにはこのようなしくみがなく，透過画像は表現できない．しかし，このK＝2のエスケープを画像データの中に埋め込んでおけば，ある範囲のドットを表示せずにとばすことができるので，透過画像が表現できそうだ．実際には，画像表示ツールがK＝2のエスケープに対応していないとだめだが….

〈図3-4〉8ビットのRLE圧縮



されたデータはすべてWORD（2バイト）境界にしたがって並びます．

画像データに同じ値が連続せず，1ピクセルごとに値が変わるとき（ラン長＝1）は，エンコード・モードを使うと各ピクセルを2バイトにエンコードするため，データ・サイズは元データの2倍に増えてしまいます．それを避けるために絶対モードが用意されていますが，絶対モードではデータ・サイズは元データより小さくなりません．

このように，RLE方式は同じパレット値が連続するデータに対しては圧縮の効果がありますが，自然画像に対してはあまり効果はありません．逆にファイル・サイズが大きくなってしまう場合もあります．

● 4ビット/ピクセルのデータの圧縮 BI_RLE4

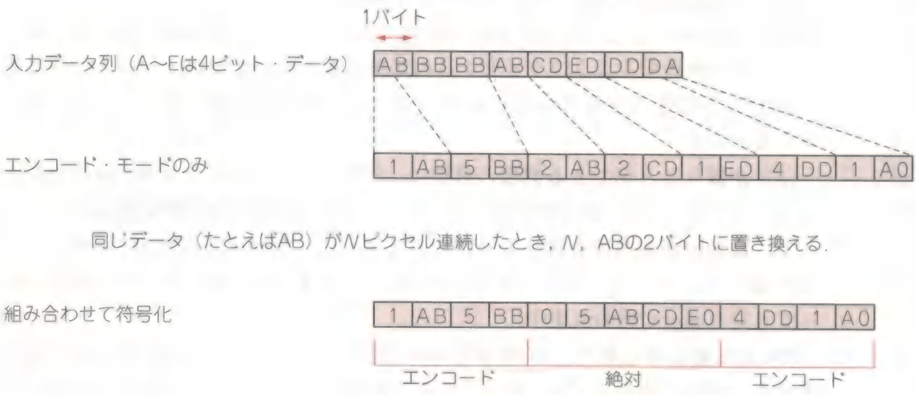
4ビット/ピクセルのデータに対して，2ピクセル（1バイト）単位でラン・レングス方式の圧縮を行います．やはり，データの圧縮を行うエンコード・モードと，圧縮しない絶対モードを混在させることができます（**図3-5**）．

エンコード・モードは2バイトの組からなり，1バイト目がピクセル数N（$N=1\sim255$），2バイト目が2ピクセル分のデータを示します．画像データに同じ2ピクセルのデータ・パターンがNピクセル分連続したとき，元データは$N/2$バイトです．それを，エンコード・モードでは2バイトに圧縮できます．たとえば“0xA，0xB”のパターンが7ピクセル連続した場合，元データは“0xAB, 0xAB, 0xAB，0xA”の3.5バイトですが，圧縮後は“0x07，0xAB”の2バイトになります．同じ値がNピクセル連続した場合も同様で，たとえば“0xA”が7ピクセル連続した場合，元データは“0xAA，0xAA，0xAA，0xA”の3.5バイトですが，圧縮後は“0x07，0xAA”の2バイトになります．

また，エンコード・モードにはBI_RLE8と同じ3種類のエスケープが定義されています．

絶対モードは，Mピクセル分の圧縮しないデータ列（$M/2$バイト）の前に2バイトを付加します．1バイト目は常に0，2バイト目はピクセル数M（$M=3\sim255$）を示します．さらに，データ列の部分のバイト数が偶数になるように，必要に応じて末尾に1〜3ピクセル分の0を付加します．これによって，圧縮されたデータはすべてWORD（2バイト）境界にしたがって並びます．

〈図3-5〉4ビットのRLE圧縮



第1-4章　コンピュータ間で画像ファイルをやりとりする画像交換フォーマット

GIF

宮崎　仁

〈名称〉

Graphics Interchange Format

〈現行バージョン〉

Ver. 89a（1989年7月）

〈仕様書〉

Graphics Interchange Format Version 89a Programming Reference (modified)

〈発行日〉

1995年1月

〈発行者〉

CompuServe Inc.

〈情報入手先〉

http://www.compuserve.com/

〈参考文献（英文）〉

T. Welch, "A Technique for High-Performance Data Compression", IEEE Computer, 1984.6

パソコン通信の時代からもっとも広く使われている画像ファイル形式

GIF（ジフ）はGraphics Interchange Format（画像交換フォーマット）という名称の通り，ネットワークを通じてさまざまなコンピュータの間で画像ファイルをやり取りするための標準フォーマットとしてもっとも普及しているものの一つです．GIFが登場した当時はパソコン通信の全盛期であり，そのデファクト・スタンダードとして広く普及しました．その後，パソコン通信に代わってインターネットが全盛となりましたが，インターネットの管理組織であるW3C（World Wide Web Consortium）では，Web上での標準画像フォーマットとしてGIFとJPEG（JFIF）を採用しました．GIFの仕様は現在のパソコンの環境から見ればかなり古く，また後述するようにライセンスの問題も抱えていますが，今のところはまだJPEG（JFIF）とともにもっとも広く用いられている画像交換フォーマットと言えるでしょう．

パソコン通信
インターネットが普及する前に盛んだったコンピュータの通信ネットワーク．1台のホスト・コンピュータに，各ユーザが個別に電話をかけてアクセスする方式をとっていた．別のホスト・コンピュータにつなぎたければ，電話を切ってかけ直さなければならないのがインターネットとの大きな違い．インターネットなら，すべてのホスト・コンピュータ（Webサイト）が常時ネットワークでつながれていて，サイトからサイトへ自由に移動することができる．

GIFが扱うことができる画像の種類は，2色（1ビット/ピクセル）から256色（8ビット/ピクセル）までのパレット・カラーです．2色に白と黒を選ぶことによって，白黒の2値画像も扱うことができます．GIFファイルにカラー・テーブルを内蔵することによって，パソコンやOS，ビデオ・カードなどの表示環境が違っても同じ表示が得られます．

256色を超える色数や，RGBなどのコンポーネント・カラーは扱うことができません．また，画像サイズ（ピクセル数）を定義するパラメータが2バイトのため，最大画像サイズは64K×64Kピクセルに制限されます．印刷に使うような

多色，高解像度画像を扱うことはできません．

GIF では画像データに対してかならず LZW 方式の圧縮を行います．LZW 方式は，単純なアルゴリズムを用いて，2 値画像から自然画像までさまざまな種類の画像データを可逆的かつきわめて高効率に圧縮できる優れた圧縮方式として知られています．GIF を利用すれば，他の可逆圧縮を利用する画像フォーマットや無圧縮の画像フォーマットにくらべて，ファイル･サイズを小型化できる場合が多いでしょう．

さらに，飛び越し表示によって画像全体の概要をすばやく表示できるインタレース GIF，背景を透明にして重ね表示のできる透過 GIF，一つのファイルに複数の静止画像を収めて連続表示ができる GIF アニメーション，画像に表示されないテキスト情報をファイルに含めることができるコメント拡張，画像とともに表示されるテキスト情報をファイルに含めることができるプレーン･テキスト拡張などの豊富な機能をもっているのも GIF の大きな特徴です．

しかし，GIF が扱える画像は最大 256 色であり，RGB や YCbCr，CMYK などのコンポーネント･カラーに対応していないというのは，現在のパソコン環境からすればやや古いと言わざるを得ないでしょう．また，LZW 方式は TIFF などいくつかの画像フォーマットで採用されていますが，Unisys 社が特許を保有しています．有料，無料を問わず，圧縮/再生ソフトウェアの開発や頒布にはライセンスが必要であり，またライセンスを受けていないソフトウェアで生成した画像ファイルの利用にも制限が加えられます．

これらの問題から，最近では GIF に代わる新しい画像フォーマットとして PNG (Portable Network Graphics) が提唱されています．

ファイル・サイズを小型化できる

GIF のファイルが小型になる理由は二つあり，一つは LZW の圧縮効率が高いこと，もう一つはフルカラーの画像が 256 色に減色されることである．フルカラー画像は 1 ドットあたり 8 × 3 = 24 ビットのデータをもつが，256 色のパレット・カラーなら 1 ドットあたり 8 ビットなので，それだけで 1/3 に圧縮できることになる．

Unisys 社

もともと LZW の特許を所有していたのは，開発者の T. Welch が在籍していた Sperry 社である．Sperry 社は Burroughs 社と合併して現在の Unisys 社になった．

画像交換フォーマット GIF の誕生

1980 年代には，パソコンやモデムが急速に普及し，ホスト･コンピュータを介してさまざまなパソコンでデータをやり取りするパソコン通信が盛んに行われていました．最初の頃は，電子メールや BBS (電子掲示板) などおもにテキスト･ベースのデータ交換を行っていましたが，次第に実行プログラムや画像データなどのバイナリ･データ交換の需要が増えてきました．

DOS など当時の代表的なパソコン OS の多くはテキスト･ベースで，画像を表示する機能をもちませんでした．そのため，パソコン上で画像を表示したいユーザは画像ローダなど個別のアプリケーションを利用していました．アプリケーションの多くは独自のデータ形式をもち，同じプラットフォーム上でさえアプリケーションが違えば画像データの交換は困難でした．

そのような中で，アメリカの大手商用 BBS である CompuServe 社では，機種やアプリケーションに依存しない画像ファイル交換フォーマットとして GIF を開発しました．GIF の最初のバージョンは 1987 年 5 月に発表された Ver. 87a (GIF87a と略記されることが多い) です．

CompuServe 社では，自社 BBS のユーザに GIF の利用を推奨するだけでなく，世界的なデファクト･スタンダードとして GIF を普及させようと考え，仕様書を公開するとともに無償でライセンスを供与してきました．GIF の登場は多くのユーザに歓迎され，幅広く普及しました．LZW 圧縮，インタレース GIF，GIF アニメーションを可能にするマルチイメージ GIF などの主要な機能は，この GIF87a にすでに含まれています．

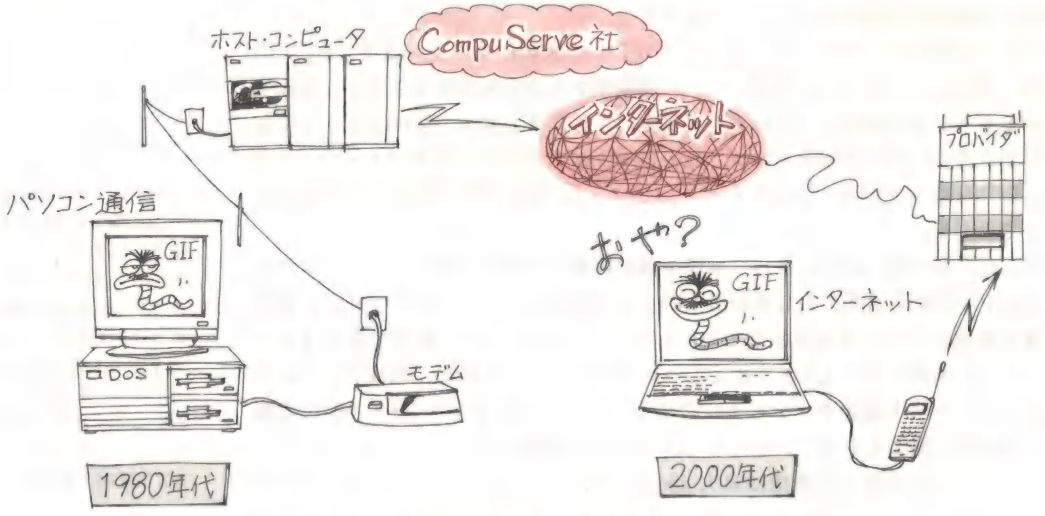
次いで，CompuServe 社では 1989 年 7 月に機能拡張版の Ver. 89a (GIF89a)

画像ローダ

当時のパソコンで画像を扱うのは一部のユーザに限られていて，市販の画像ローダもほとんどなく，フリーウェアが盛んに作られていた．

大手商用 BBS

アメリカでは CompuServe，日本では PC-VAN，ASCII-NET，NiftyServe などが有名だった．



≪GIF はパソコン通信の頃から使われている…≫

を発表しました．この GIF89a は GIF87a に対して上位互換であり，さらに透過 GIF，GIF アニメーションをサポートする**遅延表示**の規定，コメント拡張，プレーン・テキスト拡張，アプリケーション拡張などの重要な機能が追加されました．

それ以来 GIF の仕様には変更はなく，仕様書の内容も同一です．ただし，LZW 方式の特許に関する記述の追加が必要になったため，1995 年 1 月に改訂版が発行されました．

遅延表示
画像を 1 枚ずつ時間をおいて表示していくやり方．この機能がないと，アニメーションは瞬時に終わってしまう．

GIF のライセンス問題と新しい画像ファイル・フォーマット PNG

このように，GIF は CompuServe 社の努力や LZW 方式による圧縮効率の高さなどによって，画像ファイル・フォーマットのデファクト・スタンダードとして普及しました．さらに，W3C によって **html 標準の画像フォーマット**として採用されたことから，インターネットでも GIF が広く用いられてきました．

しかし，パソコンの画像表示能力は年々向上し，16 ビット/ピクセルのハイ・カラー（65 万色）や，24 ビット/ピクセルのフルカラーもしくはトゥルー・カラー（1677 万色）も一般に用いられるようになりました．その中で，最大 256 色という GIF の制限に不満をもつユーザも増えてきました．

さらに，LZW 方式を採用したことから大きな問題が起こってきました．LZW 符号は 1978 年に A. Lempel, J. Ziv の二人が開発した LZ78 符号を改良したもので，1984 年に T. Welch によって発表されました．その当時は特許ライセンスの必要がない公知の技術と思われており，GIF87a（1987 年），TIFF5.0（1988 年）などに採用されました．しかし，実際には Sperry 社（その後 Burroughs 社と合併して Unisys 社になる）が 1985 年に LZW 方式の特許を取得していました．

以前はライセンスについて事実上黙認の形となっていましたが，1993 年ごろから Unisys 社は CompuServe 社などとライセンス交渉を始め，1995 年ごろからは有料，無料を問わず GIF を扱うすべてのアプリケーションがライセンスを求められるようになりました．また，ライセンス未取得アプリケーションのユーザに対しても，作成した GIF ファイルの頒布や Web への公開に**ライセンスが課されます**．

html 標準の画像フォーマット
Web ページを記述した html 文書の中に gif ファイル名を書き込んでおけば，ページのその場所に画像が表示される．jpeg ファイルでももちろん同じことができる．これらの画像は標準として認められているので，画像を表示できる環境ならどんなブラウザでもちゃんと見える．しかし，bmp ファイル名を書き込むと，その画像は見える人にしか見えない．

ライセンスが課されます
実際に，GIF 画像を Web ページで公開していた運営者に対して，ライセンス料の支払いを求める訴訟が何件か行われている．

小規模メーカや個人が開発，頒布するアプリケーション（Webブラウザ，画像処理ツールなど）では，GIFへの対応をやめるものが増えています．一方，Microsoft社をはじめとする大手メーカの多くは，Unisys社と正規に契約を結んだ上でGIFへの対応を継続しており，ユーザは従来通りGIFを使い続けることもできます．

このような問題に対応するため，GIFに代わる新しい画像ファイル交換フォーマットとして，1995年にPNG（Portable Network Graphics）が発表されました．PNGは直接にはGIFのライセンス問題を契機として誕生したものですが，古くなってきたGIFの仕様を改良する後継フォーマットとして期待されています．

PNGはW3Cでも標準として採用されており，GIFが256色以内しか扱えないのに対して48ビット・フルカラー画像まで扱えるなど，機能的にも大幅に改良，拡張されています．ただし，古いWebブラウザや新しいものでも一部のWebブラウザはPNGに対応していないこと，アニメーション機能をもたないことなどの問題点もあります．今後は，Web上ではGIFからPNGへの移行がゆるやかに進んでいくものと思われます．

GIFファイルの定義

GIFファイルは一連の**データ・ストリーム**（GIF Data Stream）として定義されています．データ・ストリームはヘッダから始まり，全般的な表示環境を記述する論理画面（Logical Screen），データ（Data），ファイル終端を示すターミネータ（Trailer）という四つのブロックで構成されます（**図4-1**）．GIF87aとGIF89aでは用語が若干変更されていますが，ここではGIF89aの用語に従います．

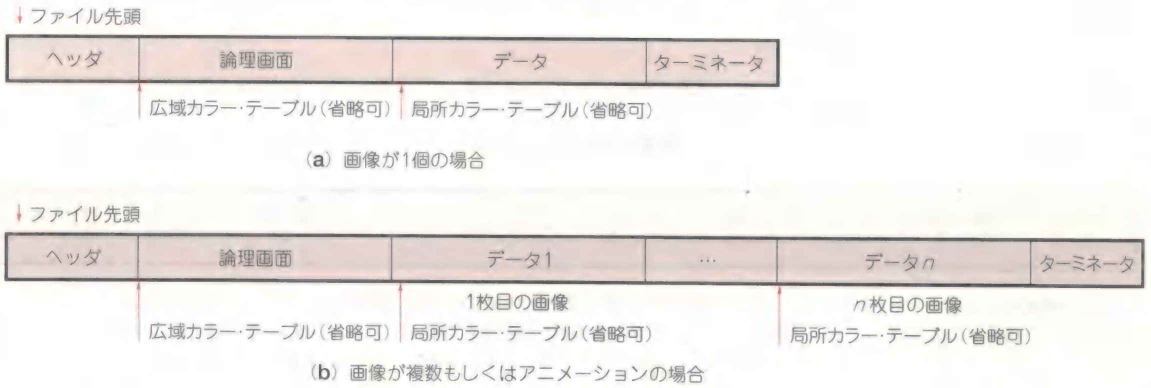
データ・ストリーム
GIFの画像データは，最初から最後まで1本につながったデータになっている．

データ・ブロックは1個だけでなく複数にできます．1個のデータ・ブロックには1個の画像のデータが含まれます．複数のデータ・ブロックを使って，1枚の論理画面の中に複数の画像を貼り付けたり，同じ位置に複数の画像を順次上書き表示することができます．さらに拡張機能として，表示の間の遅延時間を任意に設定したり，次の画像を表示するときに前の画像を塗り消すこともできます．これらの機能を利用して，GIFではアニメーションやプレゼンテーションを簡単に実現できます．

GIFで扱える画像は1ビット/ピクセル（2色）～8ビット/ピクセル（256色）のパレット・カラーです．**1ビット画像は単なる白黒ではなく**，カラー・パレットで定義した任意の2色で表示できます．カラー・パレットはデータ・ストリーム

1ビット画像は単なる白黒ではなく
白黒画像なら，ドット・データ＝'0'は白，ドット・データ＝'1'は黒が表示される（逆の場合もある）．GIFでは，ドット・データ＝'0'はパレット0に定義された色，ドット・データ＝'1'はパレット1に定義された色が表示される．

〈図4-1〉GIFファイルの構成



の中に置くことができますが，複数のカラー・パレットを柔軟に使い分けることができるのがGIFの特徴です．

画像のビット・マップ・データはかならずLZW方式で圧縮されます．無圧縮やほかの方式の圧縮は使えません．GIFファイルの作成/表示にはLZW方式の圧縮/伸長が不可欠です．

GIFの画像であることを示す文字列とバージョン情報を示すヘッダ

GIFのヘッダは6バイトの文字列で，前半の3バイトはかならず"GIF"です．

〈図4-2〉ヘッダの構成

↓ファイル先頭

'G'	'I'	'F'	'8'	'9'	'a'
識別子			バージョン		

6バイト固定長

(a) GIF89aのヘッダ

↓ファイル先頭

'G'	'I'	'F'	'8'	'7'	'a'
識別子			バージョン		

6バイト固定長

(b) GIF87aのヘッダ

〈図4-3〉論理画面の構成

ヘッダ	画面記述子 7バイト固定長，必須	広域カラー・テーブル 可変長，省略可	データ

(a) 論理画面の構成

論理画面幅（ピクセル数） 2バイト，符号なし整数	論理画面高さ（ピクセル数） 2バイト，符号なし整数	パックト情報部 1バイト	背景色 1バイト	ピクセル・アスペクト比 1バイト

論理画面幅，論理画面高さ……このGIFファイルが想定している画面のサイズ．GIFではこの論理画面の中に複数の画像を貼り付けることができる．たとえば，論理画面の中に複数の画像を同時に表示したり，少しずつ位置を変えながら順次表示することもできる．画像のサイズや貼り付ける位置は，データ・ブロックにおいて個々に指定できる．

背景色……………………広域カラー・テーブルの中で背景色として使われている色のパレット番号．広域カラー・テーブルがなければ0．

ピクセル・アスペクト比……画像のアスペクト比を1/64ステップで指定する．（この値＋15）/64が画像のアスペクト比．0のとき指定なし．

(b) 画面記述子（7バイト固定長）

Red 0 1バイト	Green 0 1バイト	Blue 0 1バイト	Red 1 1バイト	Green 1 1バイト	Blue 1 1バイト	…	Red *N*-1 1バイト	Green *N*-1 1バイト	Blue *N*-1 1バイト

パックト情報部の広域カラー・テーブル・フラグの値が1のとき，画面記述子に続いて広域カラー・テーブルが置かれる．

*N*はパックト情報部の広域カラー・テーブル・サイズの値から求められる．最大256．

(c) 広域カラー・テーブル（可変長，3×*N*バイト）

また，後半の3バイトはバージョン情報で，現在のところ“87a”または“89a”のいずれかです（**図4-2**）．きわめてシンプルなヘッダであり，GIFファイルを認識するのは容易です．

論理画面のデータ構成

論理画面は画面記述子（Logical Screen Descriptor）と広域カラー・テーブル（Grobal Color Table）の二つの部分からなります（**図4-3**）．

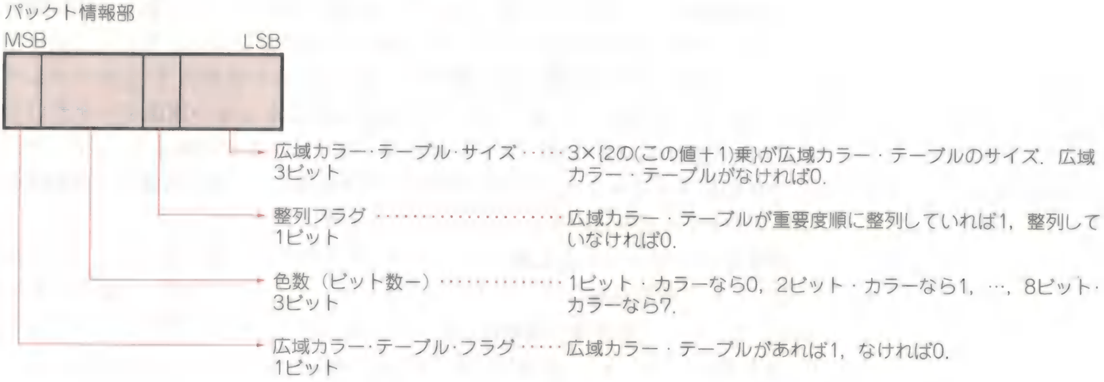
画面記述子はGIFファイルに必須の情報ブロックで，ヘッダに続けて置かれます．7バイト固定長で，画面サイズ（幅，高さ）や色数，広域カラー・テーブルがあるかどうかやテーブル・サイズなどの情報が含まれます．

広域カラー・テーブルはファイル内でデフォルトとして用いられるカラー・パレットです．広域カラー・テーブルを置くかどうかは任意であり，テーブル・サイズも可変です．R，G，B各1バイトで，表示に必要な色数分のパレット値が並べられます．カラー・パレットがN色の場合，テーブル・サイズは$3 \times N$バイトとなります．

広域カラー・テーブルがあっても，個々の画像データに局所カラー・テーブル（Local Color Table）が付属していれば，その画像データだけは局所カラー・テーブルを参照して表示されます．局所カラー・テーブルが付属していない画像はすべて，この広域カラー・テーブルを参照して表示されます．

局所カラー・テーブルも広域カラー・テーブルも含まれていないGIFファイルは，画像表示アプリケーションが以前に使用した広域カラー・テーブルのうち，

画像表示アプリケーション
画像ローダ，Webブラウザ，画像処理ツールなどが含まれる．



〈図 4-4〉データ・ブロックの構成

論理画面または前のデータ・ブロック	画像記述子 10バイト固定長，必須	局所カラーテーブル 可変長，省略可	画像データ 可変長，必須	ターミネータまたは次のデータ・ブロック

(a) 基本の構成

論理画面または前のデータ・ブロック	画像制御拡張子 8バイト固定長，オプション	画像記述子 10バイト固定長，必須	局所カラー・テーブル 可変長，省略可	画像データ 可変長，必須

(b) オプションを含む構成

画像分離子 1バイト	画像左端の位置(ピクセル数) 2バイト，符号なし整数	画像上端の位置(ピクセル数) 2バイト，符号なし整数	画像幅(ピクセル数) 2バイト，符号なし整数	画像高さ(ピクセル数) 2バイト，符号なし整数

画像分離子……………画像記述子の先頭を示す．値はつねに0x2C.
画像左端の位置…………論理画面に対する画像の左右位置を，論理画面の左端から画像の左端までのピクセル数で示す．
画像上端の位置…………論理画面に対する画像の上下位置を，論理画面の上端から画像の上端までのピクセル数で示す．
画像幅，画像高さ…………このデータ・ブロックに収められている画像のサイズ．

(c) 画像記述子（10バイト固定長）

Red 0 1バイト	Green 0 1バイト	Blue 0 1バイト	Red 1 1バイト	Green 1 1バイト	Blue 1 1バイト	…	Red *N*-1 1バイト	Green *N*-1 1バイト	Blue *N*-1 1バイト

パックト情報部の局所カラー・テーブル・フラグの値が1のとき，画像記述子に続いて局所カラー・テーブルが置かれる．
*N*はパックト情報部の局所カラー・テーブル・サイズの値から求められる．最大256.

(d) 局所カラーテーブル（可変長，3× *N*バイト）

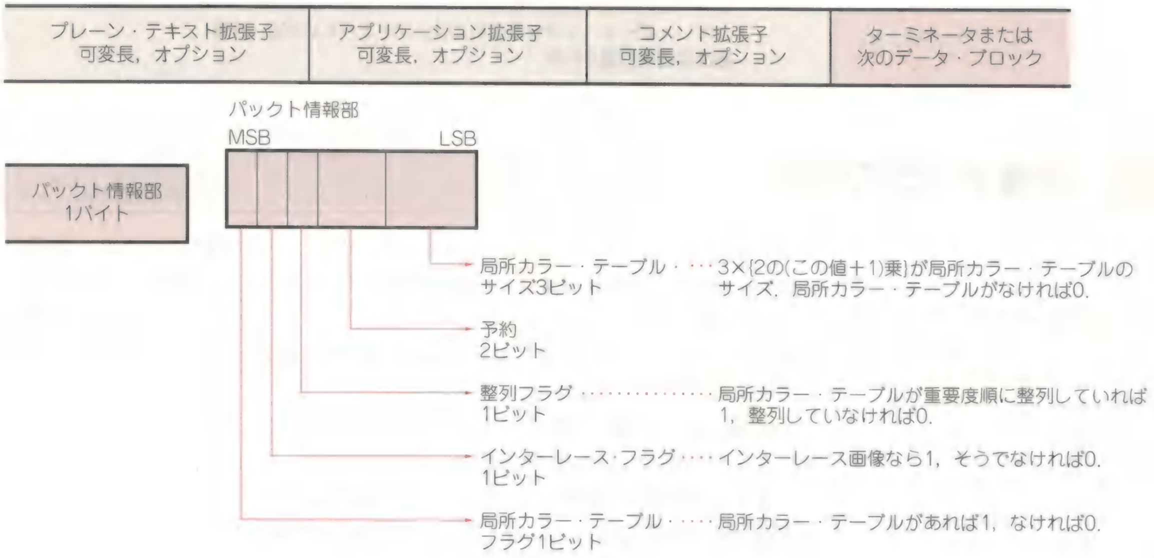
最新のものを参照して表示します．そのため，アプリケーションは最後に使用した広域カラー・テーブルを記憶しておくことが必要です．

この機能は，ネットワークなどを通じて複数のGIFファイルを配布する場合に，ファイルごとのカラー・パレットを省略して伝送量を節約するために設けられたものです．GIFの仕様では，画像データをもたない広域カラー・テーブルだけのGIFファイルや，カラー・テーブルをまったくもたない画像データだけのGIFファイルが許されます．アプリケーションでは，まず広域カラー・テーブルだけのGIFファイルを読み込み（画像は表示されない），次に画像データだけのGIFファイルを読み込めば正常な表示ができます．

局所カラー・テーブルも広域カラー・テーブルもなく，以前に使用したテーブルもないという最悪の場合は，システムのカラー・パレットを参照して表示を行います．とくに，2値画像の場合はパレットの最初の二つの値（背景色＝パレット0，前景色＝パレット1）を使用しますから，システムのパレット0には白（R＝G＝B＝0x000）を，パレット1には黒（R＝G＝B＝0xFFF）を割り当てておくように推奨されています．

データの構成

論理画面に続いてデータ・ブロックが置かれます．データ・ブロックは，基本的には画像記述子(Image Descriptor)，局所カラー・テーブル(Local Color Table)，画像データ（Image Data）の三つの部分からなります．ただし，画像記述子の前



に画像制御拡張子（Graphic Control Extention）が置かれたり，画像データの後にプレーン・テキスト拡張子（Plain Text Extention），アプリケーション拡張子（Application Extention），コメント拡張子（Comment Extention）などの**拡張情報**が置かれる場合があります（**図4-4**）．

拡張情報
これらの拡張はGIF89aで追加されたもので，GIF87aにはない．

画像記述子は10バイト固定長で，画像セパレータという1バイトのヘッダで始まり，論理画面内での画像の位置（左側，上側の余白），画像サイズ（幅，高さ），インタレース，広域カラー・テーブルがあるかどうかやテーブル・サイズなどの情報が含まれます．

局所カラー・テーブルはこのデータ・ブロック内だけで用いられるカラー・パレットで，もし存在すれば広域カラー・テーブルよりもこちらが優先されます．書式は広域カラー・テーブルと同じです．

局所カラー・テーブル
GIFファイルは複数の画像をもてるので，この画像は違うカラー・パレットで表示したいと思ったら局所カラー・テーブルを使う．全部同じカラー・パレットでいいや，というときには広域カラー・パレットを使うほうが簡単．

画像データはLZW圧縮されたビット・マップ・データです．詳しくは後に述べます．

データ・ブロックの個数は0～N個で，一つのGIFファイルに任意の個数の画像を収めることができます（マルチイメージGIF）．画像データをもたないGIFファイル（たとえば広域カラー・テーブルだけを記述したGIFファイル）を作りたい場合は，データ・ブロックは0個とします．

ターミネータの構成

1バイトで，値は0x3Bです（**図4-5**）．GIF87aではTerminaterと呼ばれていましたが，GIF89aではTrailerと名前が変わりました．

〈図 4-5〉ターミネータの構成

最後のデータ・ブロック または論理画面	終端子 1バイト

1バイト固定長，値はつねに0x3B.
ファイルの終端を示し，最後のデータ·ブロックの後に置かれる．
ただし，データ・ブロックをもたないGIFファイルでは，論理
画面の直後に置かれる．

画像データの構成

GIF では 1 ビット/ピクセルから 8 ビット/ピクセルのパレット·カラーを扱いますが，それを LZW で圧縮したものを画像データとします．元データが N ビット/ピクセルのとき，N ビットのデータ列を N ～ 12 ビットの可変長符号に置き換えます．ただし，1 ビット/ピクセルのときだけ例外で，2 ビット～ 12 ビットの可変長符号に置き換えます．

圧縮後のデータは，ふたたび 8 ビット（1 バイト）ごとにまとめて並べ直します．さらに，最大 255 バイトごとにサブブロックとしてまとめられ，サブブロックの集まりが実際の画像データになります（**図 4-6**）．

画像データの先頭 1 バイトは **LZW 圧縮の最小符号サイズ**です．その後にサブブロックの列が続きます．サブブロックの先頭 1 バイトはブロック・サイズで 0 ～ 255 の値をとります．ブロック・サイズ = 1 ～ 255 はサブブロックに含まれるデータ長を示します．ブロック・サイズ = 0 はデータを含まない 1 バイトのサブブロックで，ブロック・ターミネータとも呼ばれます．サブブロック列の終端には，かならずこのブロック・ターミネータ（0 x00）を置きます．これが，

LZW 圧縮の最小符号サイズ
LZW エンコーダは，辞書を作りながらデータを符号化していくが，辞書の N ビットの符号が満杯になったら次は $N+1$ ビットの符号を使い，$N+1$ ビットが満杯になったら次は $N+2$ ビットというようにだんだん符号が長くなる．LZW デコーダでは，圧縮データの最初は何ビットの符号が使われているかを知らないと，デコードができない．そこで，データの先頭に最初に使った符号のサイズ（すなわち最小の符号サイズ）を書いておく．

〈図 4-6〉画像データの構成

LZW最小符号サイズ 1バイト	データ・サブブロック0 可変長，最大256バイト	データ・サブブロック1 可変長，最大256バイト	…	データ・サブブロックn-1 可変長，最大256バイト	ブロック終端子 1バイト

n 個のサブブロック

LZW最小符号サイズ…LZW圧縮で使用する最小の符号サイズ．原画像が1ビット/ピクセルのとき最小符号サイズは2ビット．N ビット/ピクセル（2≦N≧8）のとき，最小符号サイズは N ビット．
ブロック終端子…1バイト固定長．値はつねに0x00.

(a) 画像データの構成

データ・バイト数m 1バイト	データ0 1バイト	データ1 1バイト	…	データ$m-1$ 1バイト

m バイトのデータ

m は1バイトの値なので，0≦m≦255の範囲となる．したがって，データ・サブブロックの長さは1～256バイト可変長．とくに，m=0のとき，サブブロックにはデータが1個も含まれない．このようなデータ・サブブロックは，ブロックの終端を示すために使われる（ブロック終端子）．

(b) データ・サブブロック

データ·バイト数 0x00 1バイト

(c) ブロック終端子

〈図4-7〉LZW圧縮のアルゴリズム

① 原画像が4ビット/ピクセルのとき，各ピクセル値は0000～1111の16通りの値をとりうる．これらを初期辞書に登録しておく．
② 1111に続く2個の符号，10000と10001は，ClearとEOIに割り当てられる．
③ 一致バッファをクリアする．
④ 入力(原画像データ)から1ピクセル分取り出し，入力バッファに入れる．このデータ(4ビット)は初期辞書に登録済み．
⑤ 一致バッファのデータの後に入力バッファのデータを結合する．
⑥ 結合したデータがすでに辞書に登録されていたら，それを一致バッファに入れて　に戻る．一致バッファは，入力から最後に取り出した登録済みパターンを保持する．
⑦ 結合したデータが未登録なら，それを辞書に登録し，一致バッファ(入力から最後に取り出した登録済みパターン)に対応する符号を出力する．
⑧ 入力バッファのデータ(初期辞書に登録済みの4ビット・データ)を一致バッファに入れて④に戻る．

4ビットのデータを複数個結合したパターンが新たに出現するごとに，辞書の未登録スペース(5ビット～)に順次登録していく(④，⑤，⑦)．
登録時は，結合する前の4ビットに対する符号を出力するので，データ圧縮にはならない(⑦)．
登録済みのパターン(8ビット，12ビット，16ビット，…)は，2回目の出現からはより短い符号(5ビット～)に置き換えられる(⑥)．同じパターンが繰り返し出現するほど圧縮の効果は大きい．
圧縮を進めるにつれて辞書に登録されるパターンは長くなり，それを短い符号に置き換えるので圧縮の効果は大きくなる．
未登録スペースがなくなったら(0xFFFまですべてパターンが登録されたら)，Clear符号を出力して辞書をクリアすることができる．
未登録スペースがなくなっても，Clear符号を出力せずにその辞書を使い続けることもできる．この場合，新たな登録はできないので，すでに登録されたパターンだけを使って符号化する．
未登録スペースが残っていても，エンコーダは任意のタイミングでClear符号を出力して辞書をクリアできる．
入力データがなくなったら，一致バッファに対応する符号を出力し，最後にEOI符号を出力して圧縮を終了する．

(a)　エンコーダの基本動作

符号(最大12ビット)	データ・パターン
0x00	0000
0x01	0001
0x02	0010
0x03	0011
0x04	0100
0x05	0101
0x06	0110
0x07	0111
0x08	1000
0x09	1001
0x0A	1010
0x0B	1011
0x0C	1100
0x0D	1101
0x0E	1110
0x0F	1111
0x10	Clear
0x11	EOI
0x12	未登録
0x13	未登録
…	…
0xFFE	未登録
0xFFF	未登録

(b)　初期辞書

入力データ列を{0x7，0x7，0x7，0x7，0x7，0x8，0x8，0x8}とする．トータル32ビット．

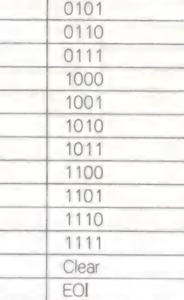
一致バッファ＝φ(③)
1個目の0x7を取り込む．入力バッファ＝0x7(④)
結合した0x7は登録済み，一致バッファ＝0x7(⑤，⑥)
2個目の0x7を取り込む．入力バッファ＝0x7(④)
結合したパターン0x77を符号0x12に登録，0x7(符号0x07)を出力，一致バッファ＝0x7(⑤，⑦，⑧)
3個目の0x7を取り込む．入力バッファ＝0x7(④)
結合した0x77は登録済み，一致バッファ＝0x77(⑤，⑥)
4個目の0x7を取り込む．入力バッファ＝0x7(④)
結合したパターン0x777を符号0x13に登録，0x77(符号0x12)を出力，一致バッファ＝0x7(⑤，⑦，⑧)
5個目の0x7を取り込む．入力バッファ＝0x7(④)
結合した0x77は登録済み，一致バッファ＝0x77(⑤，⑥)
1個目の0x8を取り込む．入力バッファ＝0x8(④)
結合したパターン0x778を符号0x14に登録，0x77(符号0x12)を出力，一致バッファ＝0x8(⑤，⑦，⑧)
2個目の0x8を取り込む．入力バッファ＝0x8(④)
結合したパターン0x88を符号0x15に登録，0x8(符号0x08)を出力，一致バッファ＝0x8(⑤，⑦，⑧)
3個目の0x8を取り込む．入力バッファ＝0x8(④)
結合した0x88は登録済み，一致バッファ＝0x88(⑤，⑥)
データ列終了，0x88(符号0x15)を出力，EOI(符号0x11)を出力．

出力符号列は{0x07，0x12，0x12，0x08，0x15，0x11}となる．
各5ビット，トータル30ビット．

圧縮後の辞書

符号(最大12ビット)	データ・パターン
0x00	0000
0x01	0001
0x02	0010
0x03	0011
0x04	0100
0x05	0101
0x06	0110
0x07	0111
0x08	1000
0x09	1001
0x0A	1010
0x0B	1011
0x0C	1100
0x0D	1101
0x0E	1110
0x0F	1111
0x10	Clear
0x11	EOI
0x12	01110111
0x13	011101110111
…	011101111000
0xFFE	10001000
0xFFF	未登録
…	…

(c)　圧縮例

トランジスタ技術 エレクトロニクスの基礎と実用技術を凝縮したフィールド・ワーク・マガジン
SPECIAL No.69
CQ出版社
B5判　172頁
定価 1,840円（税込）
特集　作ってわかる通信ネットワーク技術
携帯電話や自作LANを利用して情報収集をしよう
21世紀は通信とネットワークの時代だと言われています．現今の消費不況の中でも，携帯電話を初めとした情報通信費は伸び続けていると聞きます．今回の特集では，商用通信回線の携帯電話やパケット通信端末，計測・制御などへの利用を目的とした1線式バス・システムMicroLANなどを使って，データ収集の新しい方策を探ります．また，TCP/IPプロトコルをハードウェアで内蔵したLSIを使って，簡単な通信実験システムを作ります．


〈図 4-8〉 **LZW 伸長アルゴリズム**

①エンコーダと同様に初期辞書を生成する.
②前回符号バッファをクリアする.
③入力(圧縮符号)から符号ビット(5ビット〜)を取り出して, 今回符号バッファに入れる.
④今回符号が登録済みなら, そのパターンを出力する.
⑤前回符号のパターンに今回符号の先頭4ビットを結合して, 辞書の未登録スペースに順次登録する.
⑥今回符号を前回符号バッファに入れて③に戻る.
⑦今回符号が未登録なら, 前回符号のパターンにその先頭4ビットを結合したものを出力する.
⑧同じものを辞書にも登録する.
⑨今回符号を前回符号バッファに入れて③に戻る.

(a) デコーダの基本動作

入力符号列を{0x07, 0x12, 0x12, 0x08, 0x15, 0x11}とする. 各5ビット.
図4-7の出力符号列と同じもの

前回符号バッファ＝φ(②)
符号0x07を取り込む. 今回符号＝0x07(③).
登録済みなので, 今回符号のパターン0x7を出力. 前回符号がないので辞書には登録しない(④, ⑤).
前回符号＝0x07(⑥).
符号0x12を取り込む. 今回符号＝0x12(③).
未登録なので, 前回符号のパターン0x7にその先頭4ビットを結合. 生成したパターン0x77を出力し, 辞書(0x12)に登録(⑦, ⑧).
前回符号＝0x12(⑨).
符号0x12を取り込む. 今回符号＝0x12(③).
登録済みなので, 今回符号のパターン0x77を出力. 前回符号のパターン0x77に先頭4ビットを結合し, 生成したパターン0x777を辞書(0x13)に登録(④, ⑤).
前回符号＝0x12(⑨).
符号0x08を取り込む. 今回符号＝0x08(③).
登録済みなので, 今回符号のパターン0x8を出力. 前回符号のパターン0x77に先頭4ビットを結合し, 生成した
パターン0x778を辞書(0x14)に登録(④,⑤).
前回符号＝0x08(⑥).
符号0x15を取り込む. 今回符号＝0x15(③).
未登録なので, 前回符号のパターン0x8にその先頭4ビットを結合. 生成したパターン0x88を出力し, 辞書(0x15)に登録(⑦,⑧).
前回符号＝0x15(⑨).
符号0x11を取り込む. 今回符号＝0x11(③).
EOI符号なので伸長動作を終了.
出力データ列は{0x7, 0x77, 0x77, 0x8, 0x88}＝{0x7, 0x7, 0x7, 0x7, 0x7, 0x8, 0x8, 0x8}となる.
図4-7の入力データ列が復元された.

伸長後の辞書

符号(最大12ビット)	データ·パターン
0x00	0000
0x01	0001
0x02	0010
0x03	0011
0x04	0100
0x05	0101
0x06	0110
0x07	0111
0x08	1000
0x09	1001
0x0A	1010
0x0B	1011
0x0C	1100
0x0D	1101
0x0E	1110
0x0F	1111
0x10	Clear
0x11	EOI
0x12	01110111
0x13	011101110111
0x14	011101111000
0x15	10001000
0x16	未登録
…	…

(b) 伸長例

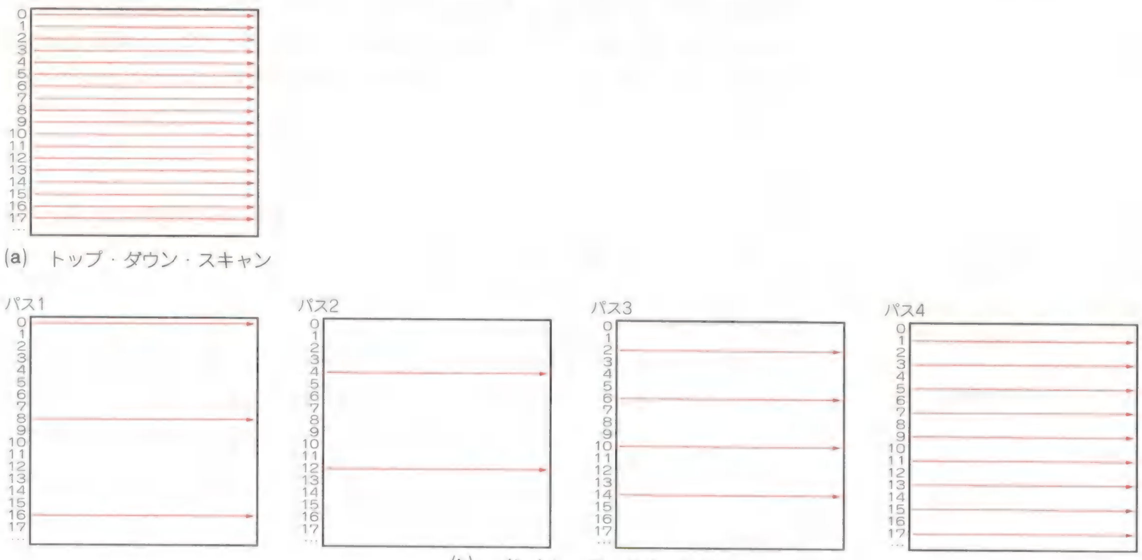
画像データの終端を示します.

LZW 圧縮の具体的な手順について GIF の仕様書には詳しい記述がなく, リファレンスとして, T. Welch, “A Technique for High Performance Data Compression” が記載されています. この標準の LZW に対して, GIF では Clear と EOI という二つの特別な符号を追加しています. Clear は辞書が満杯になったとき辞書をクリアして新しい辞書を作る機能をもちます. EOI (End Of Image) は画像データの終端を示します.

LZW 圧縮では, 入力データ列中に存在するデータ·パターンを取り出して, 符号との対応を辞書に登録していきます. そして, パターンを符号に置き換えたものを圧縮データとします (**図 4-7**). 最初は, N ビットのパターンがすべて辞書に登録されていて, それを N ビットの符号に置き換えるだけなのであまり圧縮できません. しかし, 圧縮の作業が進むにつれてより長いパターンが辞書に登録され, それを短い符号に置き換えることができるようになります.

また, 圧縮データを再生するときには, 圧縮時に使用した辞書を使わなくても, 再生側で同じように辞書を作りながら再生することができます (**図 4-8**). これは LZW 方式の優れた特徴の一つで, 圧縮データとともに辞書を受け渡す必要が

〈図4-9〉インタレースGIF



(b) インタレース・スキャン

ありません．さらに，辞書が大きくなりすぎた場合には，それまで使っていた辞書をクリアして新しい辞書で作業を続けることができます．

インタレースGIFの技法

GIFでは通常は画像を上から下に1行ずつ表示していきますが（トップダウン・スキャン），行を飛び越しながら表示することもできます（インタレース・スキャン）．画像が完全に表示されるまでの時間はどちらも同じですが，インタレース・スキャンのほうが画像全体の概要を早く見ることができます．

GIFのインタレース・スキャンは4回のパスで全画像を表示します．まず，パス1では画像の0行目，8行目，16行目，…，と8行ごとにスキャンします．次に，パス2では4行目，12行目，20行目，…，とその中間を埋めるように8行ごとにスキャンします．次に，パス3では2行目，6行目，10行目，…，とその中間を埋めるように4行ごとにスキャンします．最後に，パス4では1行目，3行目，5行目，…，とその中間を埋めるように2行ごとにスキャンします（図4-9）．

画像がインタレース・スキャンされているかどうかは，画像記述子の中のインタレース・フラグで調べます．フラグの値が1だったら表示アプリケーションでインタレース表示を行う必要があります．

画像全体の概要を早く見ることができます
ユーザに対して心理的に画像の表示が速くなったような効果を与えるだけでなく，その画像がいらないという判断が早くできるので，全部表示しないうちに表示を中止して次の動作に移ることができる．

画像制御拡張子の構成

表示後処理，ユーザ入力，透過GIF，遅延表示などの拡張機能を定義する部分です．これらの拡張機能を用いる場合は，データ・ブロックの先頭に1個の画像制御記述子を置くことができます．その後に画像記述子が続きます．

画像制御拡張子は2バイト，5バイト，1バイトの3個のサブブロックに分かれており，全体では8バイト固定長です（図4-10）．

これらの拡張機能は，表示アプリケーションに対して画像をどのように見せるかを細かく指示する機能で，GIF89a で追加されました．ほかの画像フォーマットにはあまり見られない GIF の大きな特徴と言えます．ただし，画像制御拡張子で指定しても，表示アプリケーションがその機能に対応していない場合は実行されません．

● 表示後処理

単に画像を表示するだけでなく，表示の後で画像を**背景色で塗り消したり**，以前に表示されていた画面に戻すことができます．

背景色で塗り消したり
たとえば，画面の中をキャラクタが移動していくアニメーションを作るためには，新しい画像を表示するたびに前の画像を背景色で塗り消す必要がある．アニメーション表示には必須の機能の一つ．

● ユーザ入力

画像をいきなり表示するのではなくて，ユーザの入力アクションを待って表示することができます．入力アクションとしては，特定のキーを押したり，マウスをクリックしたりということが考えられます．どのような入力アクションを用いるかは，表示アプリケーションに任されています．

● 透過 GIF

ピクセル値がパレットの特定の色（インデックス）のとき，そのピクセルの画面表示を書き換えずに元の表示を残すことができます．そこだけ画像が透けて下の画面が見えるようになります．

透過モードを使うかどうかは，**透過フラグ**によって指定します．透過フラグが 1 なら，透過インデックスで指定された値が透過色となります．透過フラグが 0 のときは，透過インデックスは無視します．

透過フラグ
透過モードを使うかどうか，かならず透過フラグで指定する．透過インデックスは常に何かの値を指しているので，透過フラグを使って透過表示を禁止しないと，常にどれかの色が透過色になってしまう．

● 遅延表示

画像をいきなり表示するのではなくて，一定時間待ってから表示することができます．遅延時間は 1/100 秒単位で，0 ～ 65535（符号なし 16 ビット整数）の範囲で指定できます．アニメーションやプレゼンテーションに利用できます．

この遅延表示とユーザ入力の両方を有効にした場合，遅延時間と入力アクションのどちらか早く発生したほうに従って表示を行います．

〈図 4-10〉画像制御拡張子

画像制御拡張子(8バイト固定長)．各データ·ブロックの先頭に置き，そこで定義されている画像の表示方法をコントロールすることができる．

拡張ブロック導入子 1バイト	画像制御拡張識別子 1バイト	ブロック・サイズ 1バイト	パック情報部 1バイト	遅延時間 2バイト，符号なし整数

ブロック・サイズのカウント範囲

サブブロック0　　サブブロック1

拡張ブロック導入子…………拡張ブロックの先頭を示す．値は常に0x21．
画像制御拡張識別子…………この拡張ブロックが画像制御拡張子であることを示す．値は常に0xF9．
ブロック・サイズ …………サブブロック1のバイト数を示す．値はつねに4．
遅延時間…………………そのデータ・ブロックの画像を再生，表示する前に，この値×(1/100)秒だけ待つ．
透過色……………………特定の色を透過にするとき(透過色フラグ＝1)，透過にする色のパレット番号を指定する．
ブロック終端子……………データ・ブロックの終端と同じ．値は常に0x00．

● GIF アニメーション

マルチイメージ GIF，表示後処理，透過 GIF，遅延表示などの機能を組み合わせることによって，GIF では効果的なアニメーションを簡単に作ることができます．

画像データの後ろにおかれる拡張情報の内容

● プレーン・テキスト拡張子の構成

箱形のテキスト・ボックスを画像とともに表示する機能です．データ・ブロック内で，画像データの直後に置くことができます．

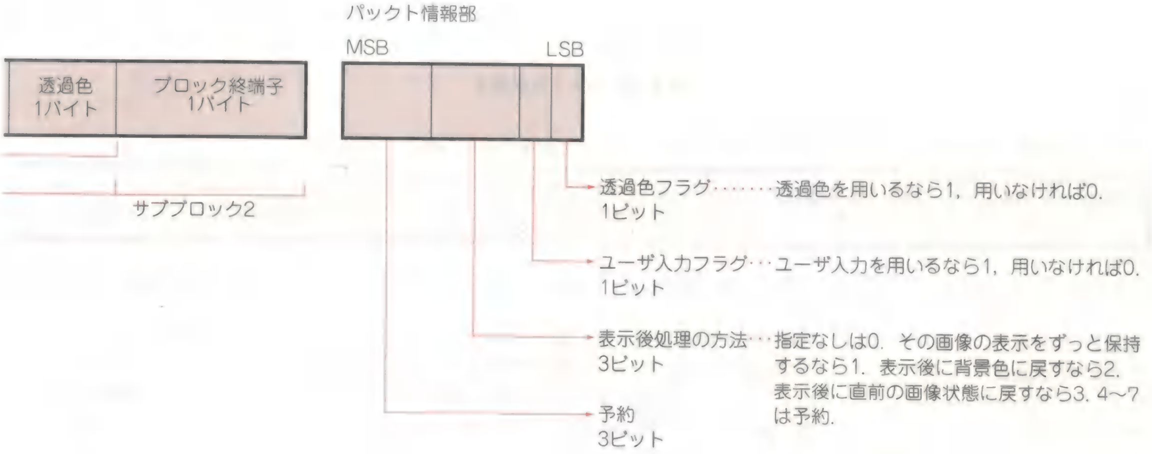
2 バイト，13 バイト，可変長のテキスト・データ，1 バイトの 4 個のサブブロックに分かれています．テキスト・データは画像データと同様に，最大 255 バイトごとにサブブロックにまとめられます（図 4-11）．

● アプリケーション拡張子

表示アプリケーション側で定義された拡張機能です．データ・ブロック内で，プレーン・テキスト拡張子に続けて置くことができます（図 4-12）．

● コメント拡張子

画面には表示されないコメント・データを画像に付加する機能です．データ・ブロック内で，アプリケーション拡張子に続けて置くことができます（図 4-13）．



〈図 4-11〉プレーン・テキスト拡張子

プレーン・テキスト拡張子（可変長）．データ・ブロック内で，画像データの直後に置くことができる．

拡張ブロック導入子 1バイト	プレーン·テキスト拡張識別子 1バイト	ブロック·サイズ 1バイト	テキスト·グリッド左端の位置 2バイト，符号なし整数	テキスト·グリッド上端の位置 2バイト，符号なし整数	テキスト·グリッド幅 2バイト，符号なし整数	テキスト·グリッド高さ 2バイト，符号なし整数

ブロック・サイズのカウント範囲

サブブロック0（2バイト）　サブブロック1（13バイト）

拡張ブロック導入子……拡張ブロックの先頭を示す．値はつねに0x21.
プレーン・テキスト拡張識別子……この拡張ブロックがプレーン·テキスト拡張子であることを示す．値はつねに0x01.
ブロック・サイズ……サブブロック1のバイト数を示す．値はつねに12.
テキスト・グリッド左端の位置……論理画面に対するテキスト・ボックスの左右位置を，論理画面の左端からテキス
テキスト・グリッド上端の位置……論理画面に対するテキスト・ボックスの上下位置を，論理画面の上端からテキス
テキスト・グリッド幅，テキスト・グリッド高さ……テキスト・ボックスのサイズ.
文字セル幅，文字セル高さ……文字セルのサイズ.
テキスト前景色……文字色のパレット番号を示す.
テキスト背景色……テキスト・ボックスの背景色のパレット番号を示す.
プレーン・テキスト・データ……テキスト・データ．画像データと同様に，最大255バイトごとにサブブロックに
ブロック終端子……データ・ブロックの終端と同じ．値はつねに0x00.

〈図 4-12〉アプリケーション拡張子

アプリケーション拡張子（可変長）．データ・ブロック内で，プレーン・テキスト拡張子の直後に置くことができる．

拡張ブロック導入子 1バイト	アプリケーション拡張識別子 1バイト	ブロック・サイズ 1バイト	アプリケーション識別子 8バイト

ブロック・サイズのカウント範囲

サブブロック0（2バイト）　サブブロック1（12バイト）

拡張ブロック導入子……拡張ブロックの先頭を示す．値はつねに0x21.
アプリケーション拡張識別子……この拡張ブロックがアプリケーション拡張子であることを示す．値はつねに0xFF.
ブロック・サイズ……サブブロック1のバイト数を示す．値は常に11.
アプリケーション識別子……アプリケーションの名称などを示す8バイトのASCII文字.
アプリケーション識別コード……アプリケーションを識別するための3バイトのコード.
アプリケーション・データ……データ．アプリケーションによって定義される．画像データと同様に，最大255
ブロック終端子……データ．ブロックの終端と同じ．値はつねに0x00.

〈図 4-13〉コメント拡張子

コメント拡張子（可変長）．データ・ブロック内でアプリケーション拡張子の直後に置くことができる．

拡張ブロック導入子 1バイト	コメント拡張識別子 1バイト	コメント·データ·サブブロック 可変長，最大256バイト	…	コメント·データ·サブブロック 可変長，最大256バイト	ブロック終端子 1バイト

サブブロック0（2バイト）　サブブロック1　サブブロック2

拡張ブロック導入子……拡張ブロックの先頭を示す．値はつねに0x21.
コメント拡張識別子……この拡張ブロックがコメント拡張子であることを示す．値はつねに0xFE.
コメント・データ……コメント・データ．画像データと同様に，最大255バイトごとにサブブロックに分割される.
ブロック終端子……データ・ブロックの終端と同じ．値はつねに0x00.

文字セル幅 1バイト	文字セル高さ 1バイト	テキスト 前景色 1バイト	テキスト 背景色 1バイト	プレーン·テキスト·データ サブブロック 可変長，最大256バイト	…	プレーン·テキスト·データ サブブロック 可変長，最大256バイト	ブロック 最終端子 1バイト

サブブロック2　サブブロック3

ト · ボックスの左端までのピクセル数で示す.

ト · ボックスの上端までのピクセル数で示す.

分割される.

アプリケーション認証コード 3バイト	アプリケーション · データ サブブロック 可変長，最大256バイト	…	アプリケーション · データ サブブロック 可変長，最大256バイト	ブロック終端子 1バイト

サブブロック2　サブブロック3

バイトごとにサブブロックに分割される.

第1-5章 ネットワークを介してやりとりする画像ファイル・フォーマット

PNG

宮崎 仁

〈名称〉
Portable Network Graphics
〈現行バージョン〉
Ver. 1.2 (1999年7月)
〈発行者〉
PNG開発者グループ
Massachusetts Institute of Technology (MIT)
〈情報入手先〉
http://www.libpng.org/pub/png/ (PNGホームサイト)
〈参考文献(英文)〉
PNG (Portable Network Graphics) Specification Ver. 1.0 (W3C Recommendation)
http://www.w3.org/TR/REC-pngで閲覧可.
http://www.libpng.org/pub/png/spec/1.0/で閲覧可.
PNG (Portable Network Graphics)Specification Ver. 1.2
http://www.libpng.org/pub/png/spec/で閲覧可.
ZLIB Compressed Data Format Specification Ver. 3.3 (RFC-1950)
ftp://ftp.isi.edu/in-notes/rfc1950.txtで閲覧可.
〈参考文献(和文)〉
PNG (Portable Network Graphics)Specification Ver. 1.2 (日本語訳)
http://tech.millto.net/~pngnews/kndh/PngSpec1.2/PNGcontents.htmlで閲覧可.

GIFフォーマットを置き換えることを最大の目的として作られた

PNGはネットワークを通じてさまざまなコンピュータの間で画像ファイルを交換するためのフォーマットとして作られました. PNGの画像処理アルゴリズムは, さまざまなコンピュータに容易に移植できるように作られており, そのために「**Portable**」という名称が付けられました. PNGが作られた直接のきっかけは, 従来パソコン通信やインターネットでデファクト・スタンダードとして用いられてきたGIF (p. 66参照) に対して, LZW圧縮の特許使用のための有償ライセンスが要求されるようになったことです.

Portable
あるプログラムがほかのマシンでそのまま実行可能, もしくは簡単な処理で移植できることをいう.

PNGは, GIFを置き換えることを最大の目的として作られており, 扱える画像の種類, 処理プログラムの小ささ, 圧縮効率, 拡張性など性能面でGIFを上回り, かつ**将来にわたってライセンスの問題を生じない**ことを目標としています. なお, PNGの読み方ですが, 「"ping"と発音する」というように仕様書に書かれています.

PNGの原案となった最初の仕様 (ドラフト1) はThomas Boutellによって書か

れ，1995年1月にcomp.graphics, comp.compression, comp.infosystems.www.providersなどのニュース・グループに投稿されました．当時これらのニュース・グループではGIFのライセンス問題について激しい議論が交わされており，この原案はたちまち大きな反響を呼び起こしました．そして，わずか数日のうちに，現在のPNGの骨格となるさまざまなアイデアが集まったと言われています．

PNGの仕様は，Thomas Boutellを中心としたニュース・グループの参加者たちによって改良が進められていきました．3月には動作可能なビュワーやPNG画像が初めて発表され，5月にはライブラリも発表されました．さらに，1995年末にはW3C（World Wide Web Consortium）の**ワーキング・ドラフト**として取り上げられ，翌1996年にはインターネットの技術委員会であるIETF（Internet Engineering Task Force）およびW3Cにおいて標準画像フォーマットの一つに承認されました．

PNGの仕様書Ver. 1.0は，1996年10月にW3Cから発行されました．編者としてThomas Boutell，協賛編者としてTom Lane，著者としてこの2人を含めた25人のメンバーが連記されています．著作権者としては，Massachusetts Institute of Technology（MIT）がクレジットされています．また，このVer. 1.0は1997年に**IETFの技術標準**RFC-0823として発行されています．

その後，マイナ・バージョン・アップとして1998年にVer. 1.1，1999年にVer. 1.2が発表されました．このバージョン・アップにはさらに複数のメンバーが加わっています．ただし，これらはまだW3C, IETFでは承認されていません．

さらに，このVer. 1.2をベースとして，現在ISO/IECにおいてPNGの国際標準化も進められています．しかし，承認までにはまだしばらく時間がかかりそうです．

将来にわたってライセンスの問題を生じない
GIFは開発元のCompuserve社がライセンス・フリーを保証し，自由に使われてきた．しかし，圧縮技術のLZWに別の会社の特許が使われていたために，後になってライセンスの問題が発生した．PNGは開発グループがライセンス・フリーを保証するだけでなく，その中で使われている個々の技術についてもライセンスの問題を生じないように注意が払われている．

ワーキング・ドラフト
標準化作業のための規格案．たたき台とでも言うべきか．

IETFの技術標準
一般にRFC標準として知られている．

PNGの普及と今後の問題点

IETFおよびW3Cの承認により，1996年の時点でPNGはGIF, JPEGと並んでインターネットにおける標準画像フォーマットとなりました．しかし，その時点では**PNGをサポートするWebブラウザ**はほとんどなく，Webサイト上にPNG画像を掲載してもほとんどのユーザはそれを見ることができませんでした．

一方，Microsoft社，Adobe社などWebブラウザや画像処理アプリケーションを供給する大手ソフトウェア会社の多くは，LZW圧縮の特許利用ライセンス契約を特許権者のUnisys社と結び，それらの会社の製品のユーザがGIF画像を従来通り掲載，閲覧できるようにしました．そのため，WebサイトからGIF画像がいっせいに消え去るという事態にはならず，GIFが急速にPNGに置き換えられるということもありませんでした．

その後，Microsoft社ではInternet Explorer 4.0から，Netscape Comunications社ではNetscape Communicator 4.04からPNG画像のインライン表示をサポートするなど，PNG対応のWebブラウザや画像処理アプリケーションも増えてきました．しかし，PNGの豊富な機能のすべてをサポートするものは少なく，基本機能だけに対応するものが多いようです．ライセンス契約を結んで従来通りGIFをサポートしている大手ソフトウェア会社にとっては，W3Cが推奨するPNGを無視することはできませんが，積極的にPNGを推進するメリットは少ないということなのでしょう．

そのため，現在ではPNGは着実に普及を続けていますが，まだGIFの方がはるかに多く使われています．

PNGをサポートするWebブラウザ
インターネットのWebページ上の画像を表示するには二つの方法がある．一つは，html文書に埋め込まれたファイル名の画像をブラウザがその場で（インライン）表示する方法だ．それには，ブラウザ自体にその画像の表示機能が組み込まれていなければならない．もう一つの方法は，html文書に埋め込まれたファイル名をプラグ・インのアプリケーションに引き渡し，アプリケーション側で表示する方法である．この場合は，ブラウザ自体はその画像の表示機能をもつ必要はない．

従来の画像フォーマットと PNG の比較

PNG は仕様面では GIF に対して大きな利点をもっています．また，JPEG や TIFF などに対しても優位な部分がありますが，劣る部分もあります．

GIF はさまざまな画像をきわめて高効率かつ可逆的に圧縮できますが，最大 256 色のパレット・カラーしか扱えません．フルカラー（24 ビット・カラー）などの画像は**減色処理**が必要であり，減色によって失われた色情報は復元することができません．一方，JPEG はフルカラーの自然画像を高効率に圧縮できますが，圧縮率を高くするほど原画像の細部の情報が失われます．また，JPEG は GIF とは逆にフルカラーとグレー・スケールしか扱えず，パレット・カラーや 2 値画像を扱うことができません．TIFF は 2 値画像からフルカラーの自然画像まで幅広く扱えますが，LZW 圧縮にはやはりライセンスの問題があり，実用上は圧縮なしで使われる場合がほとんどです．

PNG は，**最大 48 ビットのフルカラー画像**，最大 16 ビットのパレット・カラー画像，最大 16 ビットのグレー・スケール画像などを扱うことができます．これは，GIF はもちろん，JPEG や TIFF の仕様を大きく上回っています．

ただし，PNG が扱えるのは RGB 方式だけです．JPEG や TIFF のように CMYK 方式や YCbCr 方式のカラー画像を扱うことはできません．また，PNG は 1 ピクセル分の RGB データをひとまとめに扱わなければならないので，JPEG や TIFF のように R，G，B を独立プレーンとして扱うことができません．コンピュータ上で直接閲覧される画像の交換には問題ありませんが，印刷，写真などの商用の画像データを扱うには適しません．

PNG は **LZ77 をベースにした Deflate/Inflate 方式**によってさまざまな画像データを高い効率で可逆圧縮できます．さらに，画像の状態にあわせて最適な予測手法を選ぶことにより，圧縮率は LZW 圧縮と遜色ないレベルだと言われています．ただし，LZW にくらべると圧縮，伸長のアルゴリズムの実行にやや時間がかかります．また，自然画像の圧縮率は不可逆の JPEG ほど高くはできません．

インタレース表示，透過表示，プレーン・テキスト表示などの GIF の機能は，PNG ではさらに改良が加えられています．さらに，GIF にはないガンマ補正，データ・エラーを検出する **CRC** などの機能を備えています．扱える画像のサイズも $(2^{31}-1) \times (2^{31}-1)$ ピクセルと大きく，JPEG や GIF の最大 65535×65535 ピクセルを大きく上回っています．GIF を置き換えるのみならず，用途によっては TIFF や JPEG を置き換えられるだけの仕様をもっています．

とくに，フルカラー画像をなるべく画質を落とさずに扱いたい場合，従来は無圧縮の TIFF か，低い圧縮率の JPEG を選ぶしかありませんでした．このようなとき PNG を用いれば，画質を犠牲にせずにデータを効率よく圧縮できます．

一方，GIF の大きな特徴であるマルチイメージ表示や，それを利用したアニメーション GIF に相当する機能は PNG にはありません．これは，PNG の設計思想として，1 ファイルには静止画像を 1 枚だけ収めるべきであり，マルチイメージ表示は PNG の外側の機能で実現すべきだと考えたためのようです．この，複数の PNG ファイルを用いてマルチイメージ表示やアニメーション表示を行うために，MNG（Multi-Image Network Graphics）という別の仕様が作られています．

減色処理

フルカラーの画像から GIF ファイルを生成するとき，本来はあらかじめ 256 色に減色してから GIF に変換しなければならない．しかし，最近のほとんどの画像処理アプリケーションは減色と GIF への変換をひとまとめに実行してくれるので，ユーザは減色の操作をまったく意識せずに GIF ファイルを作れる．

最大 48 ビットのフルカラー画像

一般にフルカラーもしくはトゥルー・カラーと呼ばれるのは R，G，B をそれぞれ 8 ビットで表した 24 ビット・カラーである．それに対して，PNG では R，G，B をそれぞれ 16 ビットで表した 48 ビット・カラーをサポートしている．実際に使い道があるかどうかはわからないが….

LZ77 をベースにした Deflate/Inflate 方式

すでに zip, gzip, pkzip などで使われていて実績のある圧縮方式．LZW に負けないような高効率，高速の圧縮方式を新しく開発するのはとてもたいへんなので（だから LZW はライセンス料を取れるわけだ），既存の方式が選ばれた．

CRC

Cyclic Redundancy Check の略．データ通信の分野で幅広く使われている伝送エラー検出の方法．送信側では，元データを一定の計算式にかけて，計算結果を元データとともに送る．受信側では，受け取った元データを同じ計算式にかけて，計算結果を送られてきた計算結果と比較する．もし一致しなければ，どこかでエラーが発生したことがわかる．

PNG ファイルの構成

PNG ファイルは一連のデータ・ストリームとして構成されます．その点では，GIF や JPEG と似ています．PNG では，ファイル内部は**チャンク**（chunk）と呼ばれるデータ・ブロックの集まりとして定義されています．この構成は，Electronic Arts 社が作ったマルチメディア・ファイルの交換フォーマットである EA IFF 85（Standard for Interchange Format Files）に影響を受けているようです．

チャンク
もともとは，厚切りの肉やチーズのかたまりというような意味．ここでは，もちろんデータのかたまりを指す．日本語でブロックというのと同じような語感だろうか．

PNG ファイルは，先頭に置かれた PNG 識別子（signature）と呼ばれる 8 バイトのデータで始まり，その後にはすぐにチャンクが置かれ，その後にも順次チャンクが並べられていきます．PNG ファイルは，PNG 識別子とチャンクだけから構成されます．画像に関するさまざまなパラメータや，ビット・マップ・データ，カラー・パレットなどの情報は，すべて定められたチャンクの中に置きます．

チャンクはごく短いものから長いものまでいろいろなものがあり，最大長は $2^{31}-1$ バイトと定められています．チャンクのデータは基本的にバイト単位で，2 バイト整数，4 バイト整数などを扱うときは**ネットワーク・バイト・オーダ**（MSB 側を先頭に整列させる）に従います．チャンクの境界はバイト境界に揃えます．

ネットワーク・バイト・オーダ
16 ビット，32 ビットなどのデータは MSB から順に送る，という方式．したがって，ビッグ・エンディアンと同じこと．

チャンクは重要チャンク（critical chunk）と従属チャンク（ancillary chunk）に大別されます（**表 5-1**）．

重要チャンクは，IHDR（画像ヘッダ），PLTE（カラー・パレット），IDAT（画像データ），IEND（画像ファイル終了）の 4 種類だけです．このうち，IHDR，IDAT，IEND はすべての PNG ファイルに必須のチャンクです．PLTE はフルカラー画像やグレー・スケール画像には必要ないので，必須にはなっていません．ただし，フルカラー画像の場合は PLTE チャンクを置くこともできます．グレ

〈表 5-1〉PNG のチャンク

コード	従属	プラ	安全	機能	位置	必須	複数可	備考
IHDR	0	0	0	画像ヘッダ	PNG 識別子の直後	○	×	
PLTE	0	0	0	カラー・パレット	IHDR より後，IDAT より前	×	×	
IDAT	0	0	0	画像データ（圧縮済み）	IHDR，PLTE（あれば）より後	○	○	
IEND	0	0	0	画像ファイル終了	PNG ファイルの末尾	○	×	チャンク・データなし

(a)　重要チャンク

コード	従属	プラ	安全	機能	位置	必須	複数可	備考
cHRM	1	0	0	基本色度情報	PLTE，IDAT より前	×	×	
gAMA	1	0	0	ガンマ情報	PLTE，IDAT より前	×	×	
iCCP	1	0	0	埋め込み ICC 色空間	PLTE，IDAT より前	×	×	Ver.1.1 で追加
sBIT	1	0	0	有効ビット数	PLTE，IDAT より前	×	×	
sRGB	1	0	0	標準 RGB 色空間	PLTE，IDAT より前	×	×	Ver.1.1 で追加
bKGD	1	0	0	背景色	PLTE より後，IDAT より前	×	×	
hIST	1	0	0	パレット・ヒストグラム	PLTE より後，IDAT より前	×	×	
tRNS	1	0	0	透明度情報	PLTE より後，IDAT より前	×	×	
pHYs	1	0	1	物理的ピクセル・サイズ	IDAT より前	×	×	
sPLT	1	0	0	標準パレット	IDAT より前	×	○	Ver.1.1 で追加
tIME	1	0	0	最終更新時間	任意	×	×	
iTXt	1	0	1	Unicode テキスト・データ	任意	×	○	Ver.1.2 で追加
tTXt	1	0	1	Latin-1 テキスト・データ	任意	×	○	
zTXt	1	0	1	圧縮テキスト・データ	任意	×	○	

(b)　従属チャンク

〈図 5-1〉 基本的な PNG チャンク

PNG識別子 8バイト	IHDRチャンク 25バイト	IDATチャンク 可変長，複数可	IENDチャンク 12バイト

(a) カラー・パレットをもたない場合

PNG識別子 8バイト	IHDRチャンク 25バイト	PLTEチャンク (12+3×*N*)バイト	IDATチャンク 可変長，複数可	IENDチャンク 12バイト

(b) *N*色分のカラー・パレットをもつ場合

ー・スケール画像の場合は PLTE チャンクを置いてはいけません．

もっとも基本的な PNG ファイルは，PNG 識別子，IHDR チャンク，IDAT チャンク，IEND チャンクから構成されます．PNG 識別子は PNG ファイルの先頭に，IHDR チャンクは PNG 識別子の直後に，IEND チャンクは PNG ファイルの終端にそれぞれ 1 個だけ置かなければなりません．したがって，IDAT チャンクは IHDR チャンクと IEND チャンクの間に置くことになります．さらに，PLTE チャンクが存在する場合は，かならず IDAT チャンクより前に置くように決められています (**図 5-1**)．

IHDR チャンク，PLTE チャンク，IEND チャンクは 1 個しか置けませんが，IDAT チャンクは複数にすることもできます．ただし，1 個の PNG ファイルに含まれる画像はただ 1 個でなければなりません．すなわち，IDAT チャンクが複数になるのは，1 個の画像データを複数のチャンクに分割した場合です．

従属チャンクはすべてオプションで，とくに必要な場合だけファイルに付加できます．1 個しか付加できないチャンクもありますが，複数個の同じチャンクを付加できるものもあります．

ファイルにどの従属チャンクを付加するかを決める権利は，**PNG エンコーダ**がもっています．エンコーダは従属チャンクのもつ機能を活用することによって，より表示効果の高い画像ファイルを作ることができます．

一方，画像に付加された従属チャンクを利用するかどうかを決める権利は，**PNG デコーダ**がもっています．デコーダは，解釈できない従属チャンクを常に無視します．さらに，たとえ解釈できる従属チャンクであっても，それを無視することが許されます．したがって，エンコーダは従属チャンクをファイルに付加するとしても，表示側でそれが常に使われることを期待してはいけません．重要チャンクの範囲だけでも最低限の表示ができるように，画像ファイルを生成すべきです．

PNG エンコーダ
PNG ファイルを生成する能力をもつアプリケーションの総称．一般的には，画像処理アプリケーションを指す．

PNG デコーダ
PNG ファイルを表示する能力をもつアプリケーションの総称．一般的には，web ブラウザや画像表示アプリケーションを指す．

PNG 識別子のフォーマット

PNG 識別子は 8 バイト固定長かつ固定データのファイル・ヘッダです (**図 5-2**)．単にファイルが PNG ファイルであることを示すだけでなく，ネットワーク伝送や異機種間データ交換にともなうさまざまなエラーを検出したり，未然に防ぐ能力をもっています．

先頭バイトは 0x89 で，PNG ファイルをテキスト・ファイルと誤認するのを防ぐために ASCII コードでない値が選ばれています．また，インターネットなど

〈図5-2〉PNG識別子

0x89	'P'	'N'	'G'	CR	LF	ctrl-z	LF

の伝送系にはテキスト伝送時に**7ビット符号**しか扱えないものがあり，PNGファイルをテキスト·ファイルと誤認して伝送するとMSBがクリアされることがあります．PNGファイルの伝送時にこのようなエラーが発生すると先頭バイトが0x09に変化するので，受信側でエラーを検出できます．

2～4バイト目は0x50, 0x4e, 0x47です．これはASCIIコードの'P'，'N'，'G'で，ファイルがPNGフォーマットであることを示します．

5～6バイト目は0x0d, 0x0aで，これはASCIIコードのCR, LFです．

7バイト目は0x1aです．これはctrl-zで，MS-DOSなどの環境でファイルのテキスト表示を終了させる働きがあります．

8バイト目は0x0a (LF)で，5～6バイト目と合わせて，CRとLFを逆転させたりLFを除去するような不正な伝送を検出できます．

7ビット符号
たとえばASCIIコードは，0x00～0x7Fの範囲で英数字，記号，制御コードを表している．本来7ビットの符号だが，最上位に1ビットの0を付加すれば，コンピュータの1バイトに合わせて8ビットで表現することができる．

チャンクの基本フォーマットと命名規則

チャンクは基本フォーマットが統一されており，またチャンクの命名法が規定されています．これらの規則に従って作られたチャンクは，それが未知のチャンクであっても，PNGデコーダは安全にそのチャンクを無視して処理を続けることができます．

そのため，PNGでは**バージョン·アップによるチャンクの追加**が自由にできます．また，特定のアプリケーションや特定のグループだけで利用する私的なチャンク（これをプライベート·チャンクと呼ぶ）をユーザが自由に作って利用することも許されます．特定のエンコーダとデコーダの間でしか使えないチャンクがファイルに含まれていても，すべてのデコーダはファイルを正常に処理できます．もちろんそのチャンク自体は無視されますが，まったく画像を表示できないとか，処理が途中で止まってしまうということはありません．

ただし，異なるユーザが定義したプライベート·チャンクが偶然同じ名称になってしまうという危険はありえます．それを防ぐため，PNG開発者グループでは**プライベート·チャンクの登録**を受け付けています．

チャンクは，チャンク長（4バイト），チャンク·コード（4バイト），チャンク·データ（可変長），CRC（4バイト）という四つの部分から構成されます（**図5-3**）．

チャンク長は，チャンク·データのバイト数を示す**4バイト符号なし整数**です．チャンク·データ以外の部分はカウントされません．また，一般に4バイト符号なし整数は$0 \sim 2^{32}-1$の範囲の値をとることができますが，PNGでは4バイト符号なし整数を扱えない一部の処理系に配慮して，$0 \sim 2^{31}-1$の範囲（MSB＝0）に値を制限しています．これによって，チャンク長が誤って負の値として解釈されることを防いでいます．このチャンク長だけでなく，PNGファイルに含まれるすべての4バイト符号なし整数は，同様に最大$2^{31}-1$に制限されます．

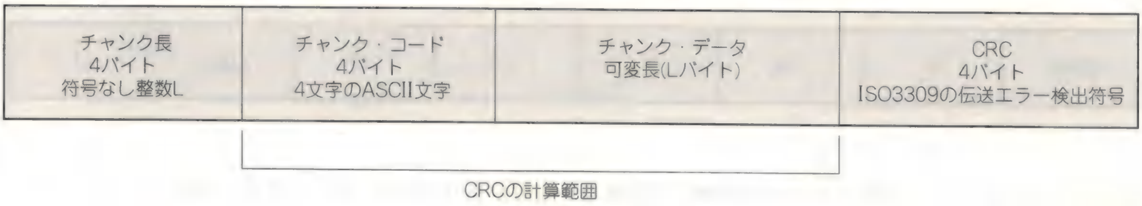
チャンク·コードは4バイトで，各バイトはASCII文字に対応します．JPEGのマーカ·コードやTIFFのタグ·コードのように単にバイナリ値がコードとして

バージョン・アップによるチャンクの追加
チャンクを追加すれば，新しい機能を定義できる．

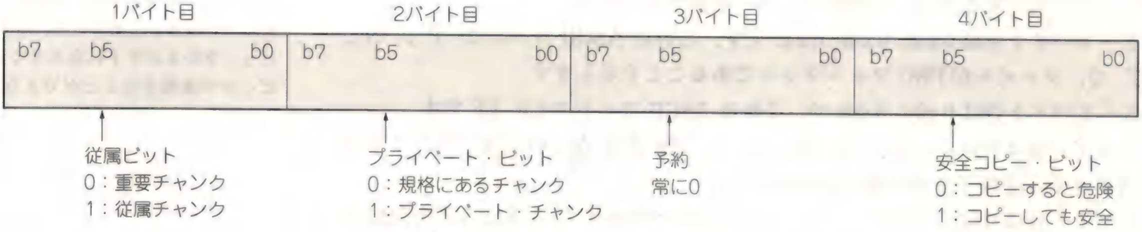
プライベート・チャンクの登録
TIFFのプライベート・タグの登録と同様のシステム．登録されたチャンク・コードを公開し，同じチャンク・コードを使わないように呼びかけるもの．

4バイト符号なし整数
実際には，4バイト符号付き整数の正の部分だけを利用している．一部の処理系では4バイト整数は符号付きしか扱えないので，それに配慮したもの．

〈図 5-3〉 チャンクの基本フォーマット



〈図 5-4〉 チャンク・コードの命名規則



ASCII文字は，0x41(A)～0x5A(Z)は大文字，0x61(a)～0x7A(z)は小文字．
b5＝0は大文字，b5＝1は小文字に対応する．

割り当てられているのではなく，人間が読んでもわかるコードになっています．

このチャンク・コードを割り当てるための命名規則が決められています（**図 5-4**）．チャンクが重要チャンクであるか従属チャンクであるか，規格上のチャンクであるかプライベート・チャンクであるか，PNG ファイルを編集するときにそのままコピーしても安全なチャンクであるかなどの情報は，コードを見るだけで判別できるようになっています．この判別用ビットは各バイトのビット 5 であり，ビット 5＝0 ならその文字は**アルファベットの大文字**（0 x41 ～ 0 x5A），ビット 5＝1 なら小文字（0 x61 ～ 0 x7A）となります．

アルファベットの大文字
これまでにでてきたチャンク・コードの IHDR, PLTE, IDAT, IEND がすべて大文字で書かれているのにはちゃんと意味があり，小文字を使ったら間違い．

1 バイト目のビット 5 は従属ビットです．重要チャンクはこのビットを 0，従属チャンクは 1 にします．したがって，重要チャンク（IHDR, PLTE, IDAT, IEND）はいずれも 1 バイト目が大文字ですが，それ以外のチャンクはすべて 1 バイト目を小文字で始めなければなりません．

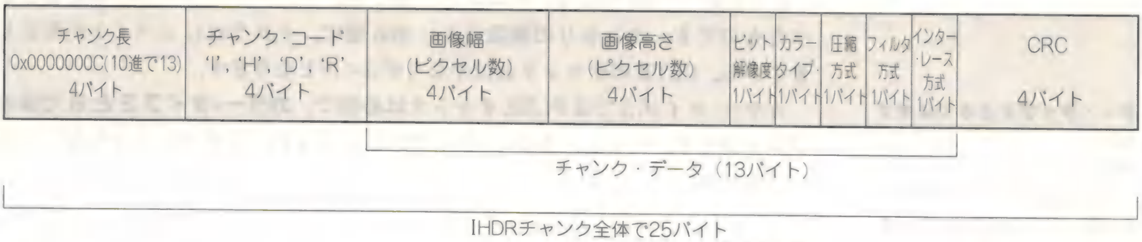
2 バイト目のビット 5 はプライベート・ビットです．規格にあるチャンクはこのビットを 0 にするので，2 文字目は大文字になります．プライベート・チャンクはこのビットを 1 にしなければなりません．

3 バイト目のビット 5 は予約です．現在あるすべてのチャンクはこのビットを 0 にします．

4 バイト目のビット 5 は安全コピー・ビットです．画像サイズや色数の変更などの画像の編集，チャンクの削除，並べ換えなどの PNG ファイルの編集を行った場合，それに合わせてパラメータを書き換えなければならないチャンクがたくさんあります．しかし，パラメータをもたないチャンクや，編集してもパラメータが変わらないチャンクもあります．編集後にパラメータを修正しないと危険なチャンクは安全コピー・ビットを 0 に，そのまま更新ファイルに書き込んでも安全なチャンクは安全コピー・ビットを 1 にします．

チャンク・データとしては，画像に関する各種のパラメータ，画像データそのもの，カラー・パレット，画像とともに表示するテキスト・データなどを扱うこと

〈図 5-5〉 IHDR チャンク



ができます．どのようなデータをとるかはそれぞれのチャンクごとに定義されています．チャンク·データの長さは0でもかまいません．

CRC は伝送で使われる一般的なもので，チャンク・コードとチャンク・データの部分だけから計算されます．チャンク長の部分は CRC 計算には含みません．

重要チャンクのパラメータ

重要チャンクは IHDR (画像ヘッダ)，PLTE (カラー・パレット)，IDAT (画像データ)，IEND (画像ファイル終了) の 4 種類です．命名規則により，重要チャンクのチャンク名 (チャンク・コード) は 4 文字とも大文字です．安全コピー・ビットが 0 なので，画像ファイルの生成，編集を行うすべての PNG アプリケーションは，すべての重要チャンクのパラメータを正しく解釈し，必要に応じて書き換える能力をもたなければなりません．

● IHDR (画像ヘッダ)

IHDR は画像の基本パラメータをまとめた 13 バイト固定長のヘッダです (**図 5-5**)．画像幅 (4 バイト)，画像高さ (4 バイト)，ビット解像度 (1 バイト)，カラー·タイプ (1 バイト)，圧縮方式 (1 バイト)，フィルタ方式 (1 バイト)，インターレース方式 (1 バイト) を情報としてもちます．

画像幅，画像高さは 4 バイト符号なし整数で，最大 $(2^{31}-1) \times (2^{31}-1)$ ピクセルの画像を扱うことができます．

ビット解像度は 1，2，4，8，16 のいずれかの値をとります．ただし，カラー·タイプによって実際にとりうる値は異なります．

カラー·タイプ 0，4 はグレー·スケール，2，6 は RGB フルカラー，3 はパレット·カラーです．また，カラー·タイプ 4，6 は各ピクセルごとに透過情報として**アルファ·データ**が付加されます．

圧縮方式は，現時点では Deflate/Inflate 圧縮 (パラメータ 0) だけがサポートされています．PNG には圧縮なしのモードはありません．

フィルタ方式は，現時点では差分予測方式による適応フィルタリング (パラメータ 0) だけがサポートされています．ただし，この適応フィルタリングは画像の特徴に合わせて 5 種類のフィルタ·タイプを選択できます．この 5 種類のフィルタ·タイプには，NONE (フィルタを使用しない) が含まれています．

インタレース方式は，現時点ではインタレースなし (パラメータ 0)，Adam7 方式インタレース (パラメータ 1) の二つがサポートされています．

アルファ・データ
各ピクセルの不透明度を表すパラメータ．このパラメータの割合に従って，ピクセルのもつ色と背景色を混色して表示する．完全に不透明なら本来のピクセルの色，完全に透明なら背景色が表示される．

● **PLTE**（カラー・パレット）

0（0x00）～255（0xFF）のエントリ値に対応する3バイトのRGB値を書き並べたものです．エントリの個数は1～256個で，それぞれに3バイトが対応しますから，PLTEのチャンク長はかならず3×*N*となります．

カラー・タイプ3ではPLTEチャンクは必須で，**カラー・タイプ2と6ではオプション**です．しかし，カラー・タイプ0と4にはPLTEチャンクを付加してはいけません．

カラー・タイプ2と6ではオプション

IHDRの中には，カラー・パレットが存在するかどうかを示すフラグがない．一つずつチャンクを調べていって，PLTEチャンクがあるかないか探す．

● **IDAT**

画像データは1行ごとに1個のスキャン・ラインとして扱われます．各スキャン・ラインは1行分のピクセルのサンプル値が連続したものです．フルカラー画像の場合，1ピクセルのデータはRGBの順で並んだ3個のサンプル値で表されます．スキャン・ラインがバイトの整数倍でないときは，最終バイトのLSB側に空きを作り，スキャン・ライン境界をバイト境界に合わせます．この空きには，0または1を任意に詰めることができます．

各スキャン・ラインの先頭に，フィルタ・タイプを表す1バイトが付加されます．フィルタ・タイプ＝NONE（フィルタを使用しない）であっても，この1バイトはかならず付加されるので注意が必要です．この操作は面倒でデータ量も増えますが，行ごとに適切なフィルタを選択すれば圧縮効率を大きく向上させることができます．

フィルタの選択はPNGの圧縮効率に大きな影響を与えるため，作成されたPNGファイルのサイズはエンコーダの性能によってかなり変わると言われています．

フィルタの選択

規格書の付録の部分では，パレット・カラーや8ビット未満のグレー・スケールには通常フィルタを用いない（フィルタ・タイプ＝NONE）方がよく，8ビット以上のグレー・スケールやフルカラーには適切なフィルタが効果をもつ可能性があると述べている．また，固定フィルタを用いるならPeathフィルタ（フィルタ・タイプ＝4）が平均的によいが，最良の結果を得るにはスキャン・ラインごとにフィルタを選択する最適化フィルタの手法がもっともよいようだ．実験的には，5種類のフィルタをそれぞれ使用してフィルタ出力の絶対値の総和を計算し，その値が最小となるフィルタを使うとよい結果が得られる，とも述べている．

各スキャン・ラインに対して，エンコーダは圧縮の前処理としてこのフィルタを用いて適応フィルタリングを行い，それからDeflate/Inflate方式の圧縮を行います．この圧縮方式では，圧縮後のデータはバイト境界に並びます．

通常は，圧縮後のデータを一括して1個のIDATチャンクとします．しかし，圧縮後のデータを任意の位置（バイト境界）で分割して，複数のIDATチャンクに収めることもできます．IDATチャンクは1個にまとめる方がファイル・サイズは小さくできますが，エンコーダの動作メモリが少ない環境では，適当なサイズに分割して扱うほうが処理が容易です．

なお，IDATチャンクが複数に分割された場合，それらはファイルの中に連続して置かなければなりません．中間に別のチャンクが挿入されると，動作は保証できません．

● **IEND**（画像ファイル終了）

IENDチャンクはチャンク・データをもたないチャンクであり，チャンク長は0です．

従属チャンクの構成

表5-1のように，現在の規格に含まれている従属チャンクには，tRANS（透明度情報），gAMA（ガンマ情報），cHRM（基本色度情報），sRGB（標準RGB色空間），iCCP（埋め込みICC色空間），iTXt（Unicodeテキスト・データ），tTXt（ISO/IEC Latin-1テキスト・データ），zTXT（圧縮されたISO/IEC Latin-1テキスト・データ），bKGD（背景色），pHYs（物理的ピクセル・サイズ），sBIT（サン

プル値の有効ビット数)，sPLT(標準パレット)，hIST(パレット・ヒストグラム)，tIME(画像の最終更新時間)があります．

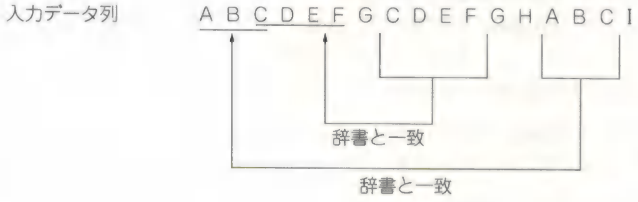
これらのうち，gAMAはVer. 1.0から存在していますが，Ver. 1.1で一部の定義が変更されました．また，iCCP，sPLT，sRGBはVer. 1.1で，iTXtはVer. 1.2で追加された新しいチャンクです．W3Cで承認されているのはPNGのVer. 1.0だけであり，これらの新しいチャンクはまだインターネットでは標準になっていません．

なお，tTXt(ISO/IEC Latin-1)で扱えるテキストは欧文だけですが，iTXt(Unicode)を用いれば日本語を含む世界各国語のテキストを扱うことができます．

LZ77方式を改良したDeflate/Inflate圧縮

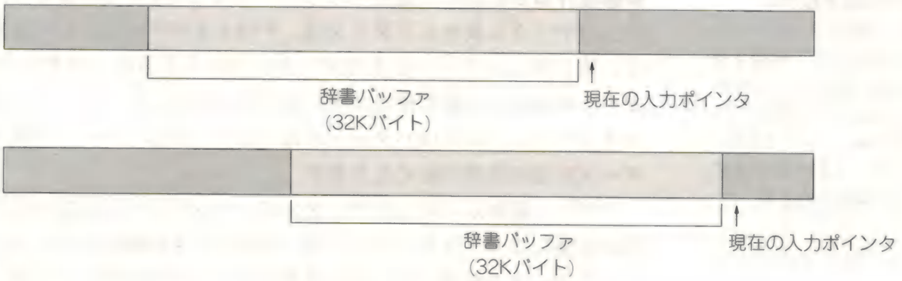
GIFやTIFFで用いられていたLZW方式に代えて，PNGではDeflate/Inflate方式のデータ圧縮を採用しています．この方式は，1977年にイスラエルのA. Lempel，J. Zivが開発したLZ77方式を改良したもので，従来zip，gzip，pkzipなどの汎用圧縮ツールで使われてきたものです．さまざまなデータを効率よく可逆圧縮できるという特徴をもっています．また，PNGの圧縮フォーマットはすでにIETFの標準として承認されているzlibフォーマット(RFC-1950)に従って

〈図5-6〉スライド窓



過去の入力データ列をバッファに記憶しておき，それを辞書とする．
2回目に現れた‘CDEFG’のパターンは，辞書の「2番目から5個」と一致するので，これをデータ(2, 5)と置き換える．
2回目に現れた‘ABC’のパターンは，辞書の「0番目から3個」と一致するので，これをデータ(0, 3)と置き換える．

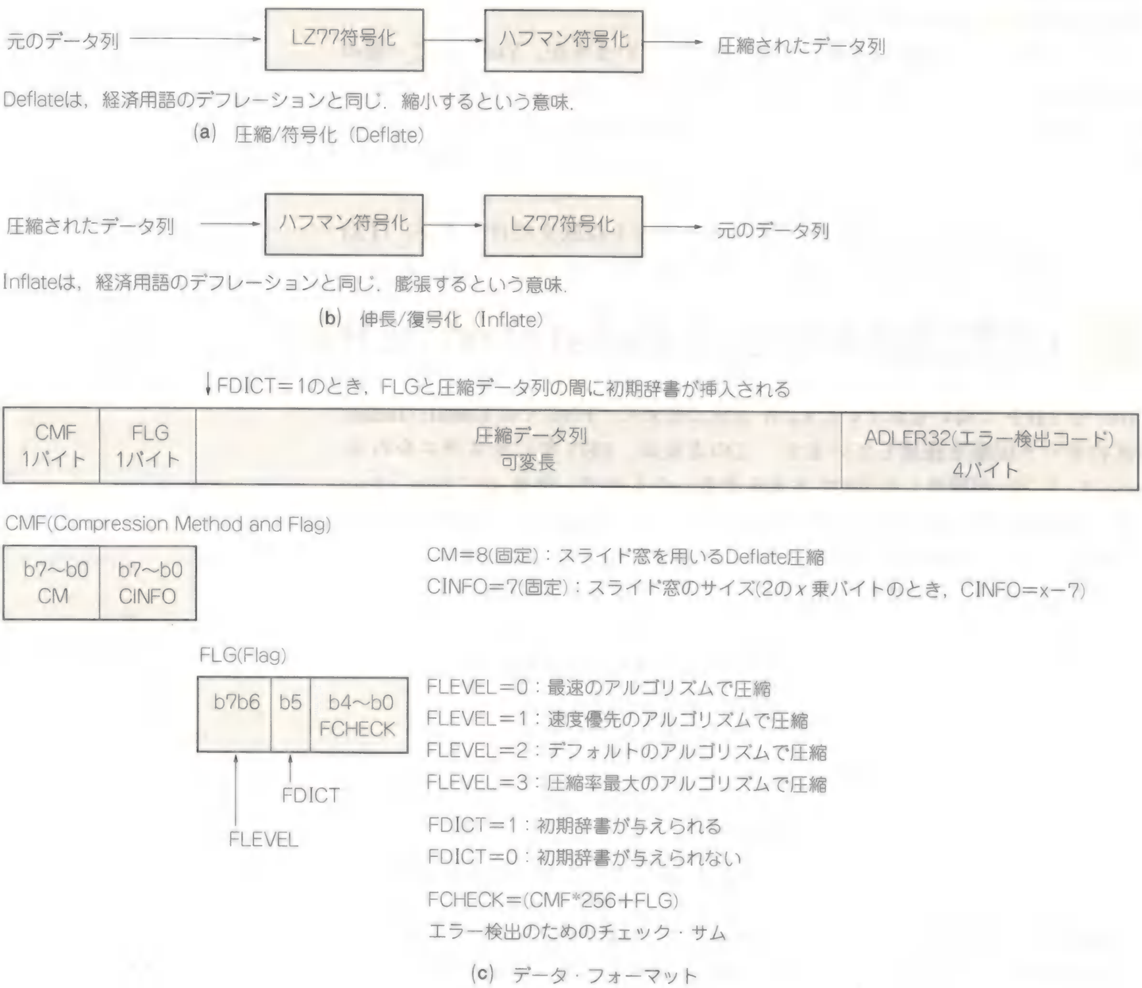
(a) 基本的なLZ77法



辞書バッファが大きくなりすぎると，メモリ・サイズが増える，辞書との比較に時間がかかる，などの問題を生じる．
これを防ぐため，現在の入力ポインタからさかのぼって32Kバイト分だけを辞書バッファに記憶し，それより前は削除する．
このような辞書バッファの構造を，スライド窓と呼ぶ．

(b) スライド窓

〈図 5-7〉 Deflate/Inflate 方式



特別な辞書を作るかわりに
実際には，過去の入力データ列そのものを辞書として利用するLZ77の方が先に作られた方式で，その欠点を改良した方式として，特別な辞書を作っていくLZ78方式が作られた．LZWは，そのLZ78をさらに改良したもの．

います．

LZ77圧縮は，LZW圧縮と同様に辞書を参照して圧縮を行いますが，**特別な辞書を作るかわり**に，過去の入力データ列そのものを辞書として使います．初めて入力データに現れたパターンは，そのままのデータとして出力されます．そして，次に同じパターンが入力データに現れたときは，パターンを（位置，長さ）という数値の組に置き換えます．位置は，最初にそのパターンが現れた位置からのオフセット，長さはパターンの長さです．同じパターンが繰り返し現れるほどデータ圧縮の効果が高くなります．

ただし，最初からの入力データのパターンをすべて記憶すると大きなメモリ空間が必要になり，またパターンの一致検出にも時間がかかります．そこで，現在の入力データから遡った一定の区間だけデータを記憶しておき，パターンの比較を行います．データの処理が進むとともにこの区間も徐々に移動していくことから，これをスライド窓と呼んでいます．PNGでは最大32Kバイトのスライド窓を使うことができます（**図5-6**）．

この，LZ77方式で圧縮したデータをハフマン符号化することにより，さらにデータを圧縮します（**図5-7**）．

第1-6章 Windows 標準のディジタル音声データ・フォーマット
WAVE

宮崎 仁

〈名称〉
ウェーブフォームオーディオ
〈発行者〉
Microsoft Corp.
〈情報入手先〉
http://www. microsoft. com/ (Microsoft 社)
〈参考資料〉
マイクロソフトウィンドウズソフトウェア開発キットマルチメディアプログラマーズリファレンス (1993年, 翔泳社)

Microsoft 社のファイル・フォーマット RIFF のサウンド・データ

WAVE (ウェーブフォーム・オーディオ) は, Windows で標準として使われているディジタル音声データ・フォーマットです. ファイルの拡張子は一般に WAV のため, WAV ファイルとも呼ばれます. WAVE では, **PCM** (Pulse Code Modulation) 音声のデータを直接扱うことができ, 音楽 CD 並みの 44.1 kHz, 16 ビット, ステレオはもとより, さまざまなサンプリング・レートやビット数, チャネル数をサポートしています.

WAVE という単独のフォーマットがあるのではなく, Microsoft 社では広くマルチメディアを扱うためのファイル・フォーマットとして RIFF (Resource Interchange File Format) という形式を定義しており, その中で WAVE のサウンド・データが扱われています. RIFF では, WAVE のほかに DIB の画像データ, PAL と呼ばれるカラー・パレット・データ, MIDI のサウンド・データなどを扱うことができます.

この RIFF は, それ以前からマルチメディア・ファイル交換フォーマットのデファクト・スタンダードとして存在していた EA IFF 85 (Standard for Interchange Format Files) の影響を強く受けています. EA IFF 85 は, コンピュータ・ゲームの有力メーカである Electronic Arts 社が, **Amiga 社**の協力を得て 1985 年に発表した規格です. ゲーム・ソフトを複数の機種に移植する際に, グラフィック・データやサウンド・データのファイルを異機種間で交換することを目的として作られました. 当時はまだ Windows はまったく普及しておらず, EA IFF 85 は MS-DOS, Macintosh, Amiga DOS をおもなターゲットとして作られました.

IFF は画像, 音声などさまざまなマルチメディア・データを扱えるように作られており, データの種類や属性を示すパラメータ, データ本体などをそれぞれチャンクと呼ばれるブロックにまとめて管理します. そして, チャンクを組み合わせてファイルを構成します. マルチメディアのファイル・フォーマットとしてはもっとも古いものの一つであり, その後に作られたさまざまなフォーマットに影

PCM
アナログ音声信号をサンプリングしてディジタル化したもの. ディジタル音声としてはもっとも単純だが, そのかわりデータ量が大きくなる.

Amiga 社
Amiga は独自のアーキテクチャをもつパソコンであり, ゲームやマルチメディア方面で根強い人気がある. ウゴウゴルーガの CG 制作に使われたのは有名. 当時は Commodore 社が製造, 販売を行っていたが, その後 Escom 社, Gateway 社と変遷してきた. しかし, Amiga の名前とアーキテクチャはまだ現役だし, さらに進化を続けている.

響を与えています．

たとえばApple社では，Macintosh上で音声を扱うための標準としてIFF互換フォーマットを採用し，AIFF（Audio Interchange File Format）と名付けました．直接互換ではありませんが，このRIFFやTIFFなどもIFFの影響を受けたフォーマットです．

WAVEは長時間の音声データを扱うのは不向き

Windowsには WAVEフォーマットのファイルを利用して音声を記録，編集，再生する機能があります．これらの機能はOSのサービスとして提供され，各種のWindowsアプリケーションから使用できます．Windows標準添付のアプリケーションには，サウンド･レコーダ（WAVEの録音再生ツール），メディア･プレーヤ（WAVE音声，AVI動画などの再生ツール）があります．また，市販のマルチメディア録音再生ツール/再生専用ツール/編集ツールなども数多く揃っています．パソコンにサウンド・ボード，マイクロホンやスピーカが備えられていれば，簡単に録音再生や編集を行うことができます．

また，**Windows起動時に流れるメロディ**（The Microsoft Sound. wav）をはじめ，Windowsの各種のメッセージや警告音はすべてこのWAVEが使われています．

Windows起動時に流れるメロディ
悪くはないが，会社などであちこちからいっせいに聞こえてくるとちょっとうんざりする．

ただし，WAVEはデータ圧縮を行わないので，このような短時間のサウンドなら良いのですが，音楽など長時間の音声データを扱うには不向きです．

WAVEフォーマットは，形式的にはWindows標準のマルチメディア･ファイル交換フォーマットであるRIFFの中の一つのタイプとして定義されています．なお，Windows標準ということから，RIFFのバイト並びはもちろんリトル･エンディアン（intel方式）です．それに対して，RIFFと同様のフォーマットでバイト並びだけをビッグ･エンディアン（motorola方式）に変えた**RIFXという形式もあります**．

RIFXという形式もあります
マニュアルにそう書いてあるのだが，実際に見たことはない．

WAVEが扱うことができるデータは，現在のところPCM方式でディジタル化された音声データ･ストリームだけです．このPCM方式は音楽CD（CD-DA）の規格と互換性があり，音楽CDのデータをWAVEファイルに変換したり，WAVEファイルから音楽用CDプレーヤで聴けるCD-Rを作ることは比較的容易にできます．

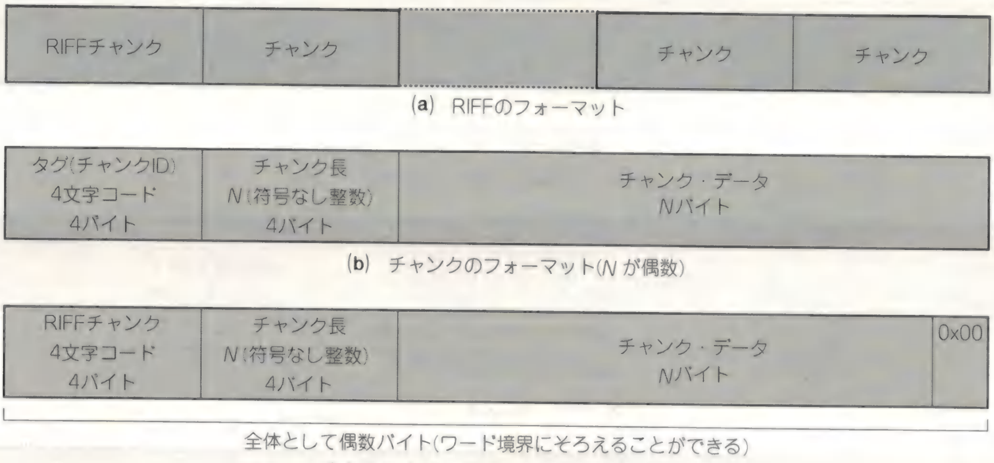
ただし，現在流通している各種のマルチメディア･アプリケーションは，正規のWAVEフォーマットだけでなくさまざまな変種フォーマットに対応したものが多く，しかも拡張子がWAVのままの変種サウンド･データも広く流通しています．たとえば，MP3の圧縮サウンド･データをWAVEの変種フォーマットに収めたようなものも使われているようです．

RIFFのフォーマットの構成

RIFFはチャンクと呼ばれるブロックの組み合わせでできています（**図6-1**）．

チャンクの先頭にはタグもしくはチャンクIDと呼ばれる4バイトのコードがあり，チャンクの種類を示します．この4バイトは4個のASCII文字を表すもので，タグはすべて人間に読める4文字の並びとなっています．Microsoft社の開発環境では，FOURCC（4文字コード）という特別のデータ型を定義しています．なお，3文字以下のタグ名を使いたいときは，文字は左詰めにして，右側は

〈図 6-1〉 RIFF のフォーマット



SP (0 x20) で埋めることができます．

タグに続いて，チャンク長 (4 バイト符号なし整数)，チャンク・データ (N バイト) が置かれます．すべてのチャンクをワード境界にそろえるために，N が奇数の場合はチャンク・データの後に 1 バイトの NULL (0 x00) が付加されます．チャンク長は純粋にチャンク・データの長さ (N) だけを含みます．付加された NULL はチャンク長に加算されません．チャンク・データの形式はチャンクの種類によって決まっており，単純なデータの場合もあれば，複数の**サブチャンク**を中に含む場合もあります．

チャンクの中に，RIFF チャンクと呼ばれる特別のチャンクがあります．これは，タグ = 'RIFF' によって始まります．RIFF チャンクは常に RIFF ファイルの先頭に置かれるので，すべての RIFF ファイルは 'RIFF' という 4 文字で始まることになります．

RIFF チャンクが特別であるというもう一つの理由は，RIFF チャンクの具体的な機能は**フォーム・タイプ**と呼ばれる 4 文字コードで定義されることです．ほかのチャンクは，タグによってチャンクの具体的な機能が決まっています．

フォーム・タイプ・コードは，RIFF チャンクのチャンク・データの先頭 4 バイトを占めます．すなわち，'RIFF'，チャンク長に続く 4 バイトが，フォーム・タイプを示す 4 文字コードになっています．フォーム・タイプとしては，'PAL' (カラー・パレット，3 文字なので後に 1 個のスペースが挿入されることに注意)，'RDIB' (デバイス独立ビット・マップ)，RMID (MIDI サウンド・データ)，RMMP (マルチメディア・ムービー)，WAVE (ウェーブフォーム・オーディオ) などが定義されています．

RIFF チャンクはただ 1 種類ですが，実際には，フォーム・タイプの数だけ異なる種類の RIFF チャンクがあると考えられます．さらに，RIFF ファイルの残りの部分 (フォーム・タイプ・コードの後) の形式も，フォーム・タイプによって決まります．すなわち，フォーム・タイプは RIFF チャンクの機能を決めるだけでなく，RIFF ファイル全体の構造も定義しています．フォーム・タイプによっては，RIFF チャンクの中に複数のサブチャンクを内包できるものもあります．フォーム・タイプごとに決められた RIFF チャンク，RIFF ファイルの構造を RIFF

サブチャンク
RIFF ではチャンクを入れ子にした構造も使われている．といっても，RIFF のチャンクには終端子がないので，あるチャンク・データに続く次のチャンクが独立したチャンクであっても，前のチャンクに含まれるサブチャンクであっても，その部分のデータの並びはまったく同じ．ではどこが違うかというと，実質的には，先頭のチャンクのチャンク長がどこまでを含んでいるかだけで見分けるしかない．

フォーム・タイプ
実際には，RIFF タグ＋チャンク長＋フォーム・タイプの 12 バイトをひとまとめにして，ひとつの RIFF ヘッダと見なす場合が多いようだ．

〈図 6-2〉 RIFF フォーム・タイプ

タグ	チャンク長	チャンク・データ	
'RIFF' 4バイト	N 4バイト	'RDIB' 4バイト	DIBフォーマットの画像データ N－4バイト

フォーム・タイプ・コード

基本のRIFF DIBファイル

(a) RIFF DIBフォーマット

タグ	チャンク長	チャンク・データ	
'RIFF' 4バイト	N 4バイト	'RMID' 4バイト	標準MIDIファイル N－4バイト

フォーム・タイプ・コード

基本のRIFF MIDIファイル

(b) RIFF MIDIフォーマット

タグ	チャンク長	チャンク・データ	
'RIFF' 4バイト	N 4バイト	'PAL' 4バイト	WindowsのLOGPALETTE構造体 N－4バイト

フォーム・タイプ・コード

基本のRIFF PALファイル

(c) RIFF PALフォーマット

タグ	チャンク長	チャンク・データ	
'RIFF' 4バイト	N 4バイト	'WAVE' 4バイト	fmtサブチャンクとdataサブチャンク N－4バイト

フォーム・タイプ・コード

基本のRIFF WAVEファイル

(d) RIFF WAVEフォーマット

フォームと呼んでいます．

実際の RIFF ファイルは，RIFF チャンクとその他のチャンクが並列に組み合わされたものもありますが，1 個の RIFF チャンク（その中にサブチャンクを内包する場合がある）だけで構成されるものも多く，RIFF チャンクが RIFF ファイルそのものであると見なせる場合も多いようです（**図 6-2**）．

フォーム・タイプはすべて Microsoft 社によって管理されていますが，ユーザが定義した新しいフォーム・タイプを Microsoft 社に登録することもできます．さらに，チャンクについても同様に，ユーザが定義した新しいチャンク・タイプを Microsoft 社に登録することができます．登録済みのフォーム・タイプ，チャンク・タイプの 4 文字コードは，**数字と大文字**だけが使われます．また，既存の登録済みタイプと重複しないコードでなければいけません．

数字と大文字

チャンク・タイプ 'RIFF' や，フォーム・タイプ 'RDIB'，'RMID'，'PAL'，'WAVE' などはすべて登録済みのタイプであることがわかる．

チャンクについては，特定のフォーム・タイプの中でだけ使われるローカルなチャンクであれば，ユーザが定義した未登録のチャンクを自由に使うことができます．この場合，4 文字コードはすべて小文字を使います．複数のフォーム・タイプで使われるグローバルなチャンクは，登録済みチャンクでなければなりません．

RIFFフォームの表記法

Microsoft社では，RIFFフォームやチャンクの構造を示すのに次のような**表記法**を用いています．たとえば，一般的なチャンクは**図6-1**のようにタグ(ckID)，チャンク長(ckSize)，チャンク・データ(ckData)の三つの部分からなりますが，これを，

<ckID>　(<ckData>)

のように表記します．チャンク長は表記から省かれます．また，チャンク・データの最後に付加されるNULLも表記から省かれます．

同様に，一般的なRIFFフォームは，

RIFF (<formType>　<sbck>. ..)

のように表記されます．<formType>はフォーム・タイプ，<sbck>はサブチャンクです．サブチャンクをもたないフォーム・タイプの場合，たとえば

RIFF ('RDIB'　data　(<DIB-data>))

のように，フォーム・タイプの後に単にデータの種類(この場合はDIBフォーマットの画像データ)を書くことができます．

さらに，これらのひとまとまりの表記に対して，

<name> → RIFF (<formType>　<sbck>. ..)

のようにしてnameというラベルを定義することができます．たとえば，'RDIB'(デバイス独立ビット・マップ)というフォーム・タイプは次のように表記されます．

<RDIB-form> → RIFF ('RDIB'　data (<DIB-data>))

あるいは，さらに読みやすくするために**改行とコメント**(//より後の部分)を加えて

<RDIB-form> → RIFF ('RDIB'　　　// RIFFヘッダ
　　　　　　　　　　data (<DIB-data>))　// DIBフォーマットの画像データ

のように表記できます．このように，比較的単純なRIFFフォームは，ほかのフォーマットで記述されたデータ全体をRIFFチャンクに収めた構造をとっており，それは元のデータ全体に

RIFF (<formType>

というヘッダを付加したものと見なすことができます．そこで，この

RIFF (<formType>

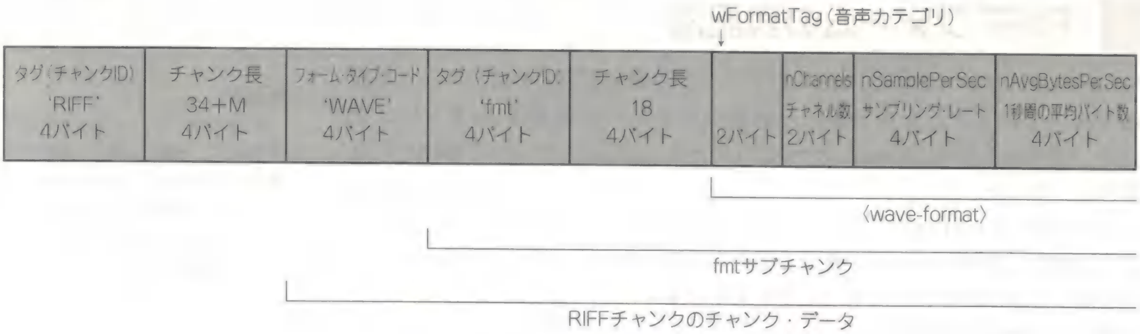
表記法
ファイル・フォーマットを定義するのに，一般には図と言葉で定義するほうがなじみがあるが，プログラマには構文で定義するほうがなじみがあるのだろう．DIBの定義に構造体を使ったりするのと同じこと．

改行とコメント
これも，ソフトウェア言語ではよく出てくる書き方である．

トランジスタ技術 エレクトロニクスの基礎と実用技術を満載したフィールド・ワーク・マガジン
SPECIAL No.67　B5判　176頁　定価1,840円(税込)
好評発売中!
ATA (IDE)，SCSI，AGPを中心とした
特集 パソコン周辺機器インターフェースⅡ

トランジスタ技術SPECIAL NO.63で「パソコン周辺機器インターフェースのすべて」という特集をしましたが，続編を望む声を多くいただいたので，今回はその続きということで，ハードディスク・インターフェースなどに使われているIDEやSCSI，高速グラフィック・アダプタに使われているAGPなどを取り上げました．さらに，前回扱いきれなかった通信関連の規格を追加して，さらに充実したかたちにしました．前回同様，リファレンスとして役立つように解説を加えました．
CQ出版社　〒170-8461　東京都豊島区巣鴨1-14-2　販売部　☎(03)5395-2141　振替00100-7-10665

〈図 6-4〉 WAVE フォーム・タイプ



(a) 構成

```
<fmt-ck>           →   fmt<wave-format>                              //wave-format 構造体
                       <format-specific >)                           //format-specific
                                                                     (音声の形式に依存する構造体，現在
                                                                     は WAVE_FROMAT_PCM だけ)

<wave-format>      →   struct{
                                   WORD wFormatTag;                  //音声カテゴリ
                                   WORD nChannels;                   //チャネル数
                                   DWORD nSamplesPerSec;             //サンプリング・レート
                                   DWORD nAvgBytesPerSec;            //1 秒間の平均バイト数
                                   WORD nBlockAlign:                 //ブロック境界
                       }

WAVE_FORMAT_PCM    →   struct{
                                   UINT nBitsPerSample;              //有効ビット数
                       }

<data-ck>          →   data (<wave-data >)                           //WAVE データ
```

(b) 表記法

〈図 6-3〉 RIFF フォームの表記法

```
〈RDIB-form〉  →   RIFF（'RDIB'                    //RIFF ヘッダ
                   data(〈DIB-data〉)               //DIB フォーマットの画像データ
```

(a) RIFF DIB フォーマット

```
〈RMID-form〉  →   RIFF（'RMID'                    //RIFF ヘッダ
                   data(〈MIDI-data〉)              //標準 MIDI ファイル
```

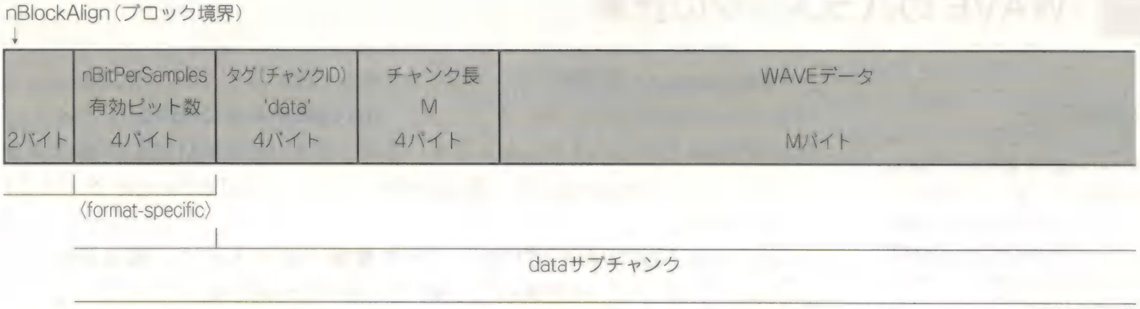
(b) RIFF MIDI フォーマット

```
〈PAL-form〉   →   RIFF（'PAL'                     //RIFF ヘッダ
                   data(〈palette:LOGPALETTE〉      //Windows の LOGPALETTE 構造体
```

(c) RIFF PAL フォーマット

```
〈WAVE-form〉  →   RIFF（'WAVE'                    //RIFF ヘッダ
                   〈fmt-ck〉                       //fmt サブチャンク
                   〈data-ck〉)                     //data サブチャンク
```

(d) RIFF WAVE フォーマット



の部分をRIFFヘッダと呼ぶことがあります．

この表記法を用いると，**図6-2**に示した各種のRIFFフォームは**図6-3**のように書くことができます．表記からは省かれていますが，タグの4文字コードの後には実際にはかならずチャンク長が挿入されることに注意してください．ただし，フォーム・タイプの4文字コードにはチャンク長は付きません．

ウェーブフォーム・オーディオのデータ構成

たとえばRDIBフォーム・タイプやRMIDフォーム・タイプの場合，data（<DIB-data>）やdata（<MIDI-data>）などの元データが，すでに必要なパラメータをすべて含んだものとなっているので，RIFFチャンクで新たにパラメータを付加する必要がありません．そのため，これらのRIFFフォームは**元データにRIFFヘッダを付加しただけの単純な構成**になっています．

しかし，WAVE（ウェーブフォーム・オーディオ）の場合はそれらと違って，**data（<wave-data>）は単なるデータだけであり**，パラメータを含んでいません．必要なパラメータはRIFFフォームで定義しなければならないので，他のRIFFフォームにくらべてWAVEは比較的複雑な構成となっています（**図6-4**）．

WAVEファイル（WAVEフォーム・タイプのRIFFファイル）は，1個のRIFFチャンクから構成され，その中に<fmt-ck>と<data-ck>という2個のサブチャンクを含んでいます．<fmt-ck>は各種のパラメータを含み，<data-ck>は生の音声データです．

<fmt-ck>サブチャンクは，**fmtという名前の未登録チャンク**です．その内容（チャンク・データ）は，<wave-format>と<format-specific>の二つの部分からなります．<wave-format>はすべてのWAVEファイルに共通なパラメータのリストであり，構造体として定義されています．<format-specific>も構造体ですが，音声データのカテゴリによって変わる固有のパラメータのリストとなっています．ただし，現在カテゴリとして定義されているのはWAVE_FORMAT_PCM（PCM音声）だけであり，実質的には<format-specific>の部分もすべてのWAVEファイルに共通です．

<data-ck>サブチャンクは，**dataという名前の未登録チャンク**です．この中にはPCM音声データが含まれます．

このほかに，**音声データに関連する補助的な情報**（作成者，ソフトウェアなど）を示す登録済みチャンクをRIFFファイル内に挿入することもあります．

元データにRIFFヘッダを付加しただけの単純な構成に
実は，それだけならわざわざRIFF形式にする必要もない．RIFF形式にするメリットは，この後にINFOリストという特別なチャンクを付加して，タイトルやアーティスト，エンジニアなどのさまざまな情報を記録できることである．INFOリストについては，RIFF WAVE形式（p. 96）を参照．

data（<wave-data>）は単なるデータだけであり
サンプリングした値が並んでいるだけである．しかも，それがバイト・データの列なのか，ワード・データの列なのか，サンプリング・レートがいくつなのかも判断できない．

fmtという名前の未登録チャンク，dataという名前の未登録チャンク
チャンク・タイプが小文字だから，これは未登録チャンクだとわかる．WAVEフォーム・タイプの中でだけ使うローカルなサブチャンク．

音声データに関連する補助的な情報
INFOリストという特別なチャンク・タイプが用意されている．

WAVEのパラメータの計算

nAveBytesPerSec
1秒あたりの平均データ・サイズ(バイト数)を表すもの．PCM音声は常にデータ・サイズは均一なので，とくに平均値をとる必要はない．ADPCM方式や，各種の圧縮方式を用いた音声データは，元の音声の状態によってデータ・サイズが変わるので，平均値をとって表す．

<wave-format>構造体のメンバは，wFormatTag (2バイト)，nChannels (2バイト)，nSamplePerSec (4バイト)，**nAvgBytesPerSec** (4バイト)，nBlockAlign (2バイト) からなります．また，WAVE_FORMAT_PCMカテゴリに対応する<format-specific>構造体のメンバは，nBitsPerSample (2バイト) からなります．

wFormatTagはカテゴリを2バイト整数で表すもので，現在はwFormatTag= 1 (WAVE_FORMAT_PCM) だけが有効な値です．

nChannelsは音声のチャネル数を2バイト整数で表すもので，nChannels= 1 (モノラル)，nChannels= 2 (ステレオ) です．3，4，…，などの値をとることも可能ですが，具体的なスピーカ配置との対応は定義されていません．

nSamplePerSecは音声をディジタル化するときのサンプリング・レート (単位Hz) を4バイト整数で表すものです．nSamplePerSec = 44100 (44.1 kHz)，nSamplePerSec = 22050 (22.05 kHz)，nSamplePerSec = 11025 (11.025 kHz) のいずれかをとることができます．44.1 kHzはCD-DAと互換です．22.05 kHzと11.025 kHzは**サンプリング・レート**をそれぞれ1/2，1/4に減らすことにより，ファイル・サイズを1/2，1/4に小さくできます．そのかわり，帯域幅も1/2，1/4に狭くなるため，音質は劣化します．

サンプリング・レート
Windowsに標準添付されるサウンド・レコーダでは，サンプリング・レート，量子化ビット数，チャネル数を選択できる．44.1 kHz, 16ビット，ステレオを「CDの音質」，22.05 kHz, 8ビット，モノラルを「ラジオの音質」，11.025 kHz, 8ビット，モノラルを「電話の音質」と呼んでいる．

nBitsPerSampleは音声をディジタル化するときの1サンプルあたりの有効ビット数を表すものです．nBitsPerSample = 16はCD-DAと同じ16ビットPCMです．nBitsPerSample = 8はISDNと同じ8ビットPCMです．実際に使われるのはこのどちらかが多いですが，それ以外にも1以上の任意の値をとることができます．

nAvgBytesPerSecとnBlockAlignの二つのパラメータは，PCM音声の場合は他のパラメータから計算して自動的に値が決まります．計算式は，

nBlockAlign = nChannels × nBitsPerSamples/8
nAvgBytesPerSec = nBlockAlign × nBitsPerSec

です．

たとえば，44.1 kHz，ステレオ，16ビットPCMのデータ (CDと同じ音質) の場合，fmtチャンクは，

fmt (1，2，44100，176400，4，16)

となります．

11.025 kHz，モノラル，8ビットPCM
サウンド・レコーダでは「電話の音質」と呼んでいる．電話局のディジタル交換機やISDNでは8 kHz, 8ビットのPCMが使われているので，実際の電話はもう少し音質が悪くなる．

また，**11.025 kHz，モノラル，8ビットPCM**のデータの場合，fmtチャンクは，

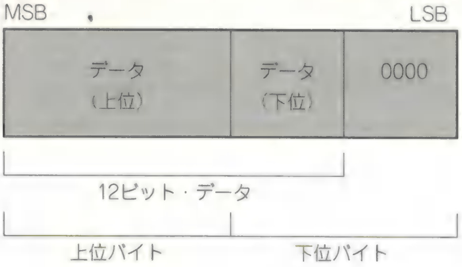
fmt (1，1，11025，11025，1，8)

となります．

WAVEのデータの構成

アナログ音声信号をサンプリングして，各瞬間の振幅に対応するディジタル値の列を生成したものです．サンプリング・レートはnSamplePerSec，ディジタル値の有効ビット数はnBitsPerSampleで与えられます．なお，実際のデータ列はバイト単位です．有効ビット数が8の倍数でないときは，データはMSB側に詰められ，LSB側の空きビットは0で埋められます (**図6-5**)．

〈図 6-5〉 1 サンプル分のデータ (n = 12 の場合)



〈図 6-6〉 データ列の並び方

サンプル1				サンプル2				
チャネル0(L)データ		チャネル1(R)データ		チャネル0(L)データ		チャネル1(R)データ		…
(下位)	(上位)	(下位)	(上位)	(下位)	(上位)	(下位)	(上位)	

←1バイト→

(a) ステレオ，16ビットPCMの場合

サンプル1	サンプル2	サンプル3	サンプル4	サンプル5	サンプル6	サンプル7	サンプル8	…

←1バイト→

(b) モノラル，8ビットPCMの場合

また，**有効ビット数によってデータ形式が変わり**ます．nBitsPerSample = 1 ～ 8 の場合，データ形式は符号なし整数（8 ビットなら 0 ～ 255）となります．nBitsPerSample ≧ 9 の場合，データ形式は符号付き整数（16 ビットなら − 32768 ～ + 32767）となります．

さらに，音声がモノラル（nChannels = 1）の場合はデータはサンプリング順にそのまま 1 列に並ぶだけですが，ステレオ（nChannels = 2）の場合はチャネル 0（L チャネル），チャネル 1（R チャネル）が交互に並びます（**図 6-6**）．

nAvgBytesPerSec は 1 秒あたりのデータ・サイズを示すもので，WAVE 再生アプリケーションに対して準備すべきバッファ・サイズを予告します．

たとえば，44.1 kHz，ステレオ，16 ビット PCM のデータ（CD と同じ音質）の場合，1 秒あたりのデータ・サイズは 176400 バイト，10 分間ではおよそ 100 M バイト余りとなります．これは，ほぼ WAVE ファイルのサイズを示すものと考えられます．

有効ビット数によってデータ形式が変わります

もともと，電話システムで使われている PCM 規格とオーディオで使われている PCM 規格ではデータ形式が違う．8 ビット以下は電話システムの仕様に合わせ，それ以上はオーディオの仕様に合わせたもの．

第1-7章 ディジタル音声データの圧縮に使われる標準フォーマット

MP3

宮崎　仁

〈名称〉

MPEG-1 Audio Layer3

〈発行日〉

1993年8月01日（ISO/IEC 11172-3）

〈発行者〉

ISO (International Organisation for Standardisation)

IEC (International Electrotechnical Commission)

〈規格〉

ISO/IEC 11172-3:1993（MPEG-1オーディオ）

〈関連サイト〉

http://www. cselt. stet. it/mpeg/（MPEGホームページ）

http://www. iso. ch/（ISO）

http://www. iec. ch/（IEC）

http://www. iis. fhg. de/（Fraunhofer IIS-A）

〈参考資料〉

SDMI Portable Device Specification Part1（Ver. 1.0）

MPEG-1とMP3

MP3はディジタル音声信号を高効率に圧縮，符号化するための規格です．音響心理学モデルに基づき，音声信号のうち聴感上で影響の少ない部分を省略することにより，**CD-DA**（音楽CD）並みの**音質を保ちながら**データを約1/11に圧縮できると言われています．圧縮は非可逆であり，完全に元のデータに戻すことはできません．

MP3はもともと，ムービーなど動画を圧縮，符号化するための国際標準であるMPEG（Moving Picture Experts Group）規格の中で，音声データの圧縮，符号化部分として作られました．動画データは画像データと音声データからなりますが，MPEG規格ではそれらを別個に圧縮/伸長する方法を採用しています．そのため，音声部分の規格だけを取り出して利用することが可能なわけです．

MPEGはその用途に応じてMPEG-1，MPEG-2，MPEG-4，MPEG-7などに分かれています（まだ規格案としてISO/IECで審議中のものも含まれる）．これらはいずれも同じMPEGという名称ですが，それぞれが独立した規格となっています．後にできた規格が前の規格のバージョン・アップというわけではなく，どの規格も並列的に存在しています．また互換性も部分的にしか維持されていません．

一言で言えば，MPEG-1はCD-ROMやHDDなどで動画データを扱うための規格，MPEG-2はDVDやディジタルTV放送などで動画データを扱うための規格，MPEG-4はインターネット上でインタラクティブに動画データを扱うための

CD-DA

Compact Disc-Digital Audioの略．もともと音楽CDが作られたときは，CDはそれ一種類しかなかったので，単にCDと呼んでいた．後になって，見た目は同じで用途やデータ・フォーマットがまったく異なるCD-IやビデオCD，CD-ROMなどが出てきたので，それらと区別したいときはわざわざDAと付けるようになった．

音質を保ちながら

画質や音質については実際に専門家が視聴して評価試験を行っているが，厳密に定量的に定義できるわけではない．広告のうたい文句と見なすべきだろう．

〈表7-1〉MPEG規格

略称	ISO/IEC番号	名称	おもな用途
MPEG-1	11172	Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbit/s	ビデオCD，MP3（音声）
MPEG-2	13818	Generic coding of moving pictures and associated audio information	DVD，ディジタルTV，MP3（音声），AAC（音声）
MPEG-4	14496	Coding of audio-visual objects	インターネット，移動体通信，ディジタルTV，AAC（音声）
MPEG-7	15938	Multimedia Content Description Interface	マルチメディア検索，マルチメディア記述
MPEG-21	18034	Multimedia Framework	マルチメディア・フレームワーク

(a) MPEGの概要

パート	ISO/IEC番号	名称
Part1	ISO/IEC 11172-1:1993	システム
Part2	ISO/IEC 11172-2:1993	ビデオ
Part3	ISO/IEC 11172-3:1993	オーディオ
Part4	ISO/IEC 11172-4:1995	試験方法
Part5	ISO/IEC TR11172-5:1998（レポート）	ソフトウェア・シミュレーション

(b) MPEG-1の規格（年のあるものは承認済）

パート	ISO/IEC番号	名称
Part1	ISO/IEC 13818-1:1996	システム
Part2	ISO/IEC 13818-2:1996	ビデオ
Part3	ISO/IEC 13818-3:1995（1998改訂）	オーディオ
Part4	ISO/IEC 13818-4:1998	試験方法
Part5	ISO/IEC TR13818-5:1997（レポート）	ソフトウェア・シミュレーション
Part6	ISO/IEC 13818-6:1998	システム拡張 DSM-CC（Digital Storage Media Command and Control）
Part7	ISO/IEC 13818-7:1997	オーディオ拡張 AAC（Advanced Audio Coding）
Part9	ISO/IEC 13818-9:1996	システム拡張 RTI（Real Time Interface）
Part10	ISO/IEC 13818-10:1999	試験方法拡張 DSM-CC（Digital Storage Media Command and Control）

(c) MPEG-2の規格（年のあるものは承認済）

パート	ISO/IEC番号	名称
Part1	ISO/IEC 14496-1:1999	システム
Part2	ISO/IEC 14496-2:1999	ビジュアル
Part3	ISO/IEC 14496-3:1999	オーディオ
Part4	ISO/IEC 14496-4（検討中）	試験方法
Part5	ISO/IEC 14496-5:2000	リファレンス・ソフトウェア
Part6	ISO/IEC 14496-6:1999	DMIF（Delivery Multimedia Integration Framework）
Part7	ISO/IEC 14496-7（検討中）	MPEG-4向け最適化ソフトウェア

(d) MPEG-4の規格（年のあるものは承認済）

パート	ISO/IEC番号	名称
Part1	ISO/IEC 15938-1（検討中）	システム
Part2	ISO/IEC 15938-2（検討中）	記述言語
Part3	ISO/IEC 15938-3（検討中）	ビジュアル
Part4	ISO/IEC 15938-4（検討中）	オーディオ
Part5	ISO/IEC 15938-5（検討中）	マルチメディア記述の方法
Part6	ISO/IEC 15938-6（検討中）	リファレンス・ソフトウェア
Part7	ISO/IEC 15938-7（検討中）	試験方法

(e) MPEG-7の規格

パート	ISO/IEC番号	名称
Part1	ISO/IEC TR18034-1（検討中）	マルチメディア・フレームワーク

(f) MPEG-21の規格

規格，MPEG-7は動画データを汎用データとして検索，管理するための規格と位置付けられます（**表7-1**）．

MP3は，1993年に成立したMPEG-1規格（ISO/IEC 11172）の中のオーディオ部分に含まれる規格です．MPEG-1の規格はPart1（システム），Part2（ビデオ），Part3（オーディオ）の3部分に分けて構成されています．さらに，MPEG-1オーディオはもっとも単純な方式のレイヤ1からもっとも複雑な方式のレイヤ3まで3段階に分けられています．MP3はそのレイヤ3にあたり，もっとも高い圧縮率が得られるかわりに，**符号化/復号化の処理ももっとも複雑です**（**表7-2**）．

ちなみに，MPEG-1オーディオのレイヤ2はMP2と呼ばれています．MP3とMP2は処理方式に互換性はありませんが，一般に流通しているMP3用アプリケーションにはMP2もサポートするものがあります．

また，MPEG-2はMPEG-1に対して多くの部分で上位互換性をもっており，**MPEG-2オーディオ**のレイヤ2，レイヤ3もそれぞれMP2，MP3と呼ばれています．これらは，MPEG-1オーディオのレイヤ2，レイヤ3の拡張仕様と考えることができます．MPEG-2はMPEG-1にくらべて画像，音声とも圧縮率が高いのが特徴です．

しかし，現在一般的に使われているMP3の仕様は，MPEG-1仕様の範囲内にとどまっており，MPEG-2による拡張を考慮する必要はありません．

符号化/復号化の処理ももっとも複雑です
とくに符号化の処理は複雑で，処理に時間がかかる．復号化，再生はリアルタイムで実行できるのが普通（できないと不便）だが，録音，符号化はリアルタイムで実行できないのが普通．一般には，いったんPCMデータ（たとえばWAVEファイル）としてディスク上に保存し，それをMP3エンコーダで「ゆっくり」符号化する．

MPEG-2オーディオ
MPEG-2オーディオでは，Layer1～3とは別にAAC（Advanced Audio Coding）と呼ばれる新しい方式も採用している．MP3以上の高音質でさらに高い圧縮率が得られることから，最近普及し始めているようだ．

MP3の普及と問題点

MPEGの規格を作ったのは，ISOとIECの合同プロジェクトとして発足した**専門委員会**です．この専門委員会がMPEG（Moving Picture Experts Group）という名称だったので，作られた規格もMPEGと呼ばれるようになりました．MPEG委員会では，MPEG規格の音声部分を単独で利用するアプリケーションを当初から想定しており，たとえば，MPEG-1オーディオを利用した高音質の録音/再生システムを早くから提案してきました．

それが一般に注目されたのは，1997年にインターネットを通じてMP3データを配信するWebサイトとしてmp3.comがスタートしたことと，MP3データを再生する携帯用プレーヤの**mpmam**がセイハン情報システム社（韓国Samsung社グループ）から発売されたことです．これをきっかけとして起きたMP3のブームは1998年中に世界中に広がり，携帯用MP3プレーヤやパソコン上で動作するプレーヤ/エンコーダ/デコーダなどのMP3アプリケーションが次々に登場しました．また，インターネットを通じて各種のMP3データを配信するWebサイトも次々に誕生しました．

MP3は音声データの圧縮，符号化の方式としてはたいへん優れたものの一つです．さまざまな音楽データを手ごろなサイズにファイル化し，HDDに保存したり，CD-RやMOなどの可搬な媒体でやり取りしたり，インターネットを通じてやり取りできます．入力データとしてCD-DAやDATなどで使われている高音質のPCM音声データをそのまま利用でき，圧縮による音質の低下もわずかです．

また，MPEG-1というすでに普及している国際標準の一部なので，既存のMPEG-1用LSIやソフトウェア・ライブラリを利用でき，中小ベンチャーや個人レベルでも圧縮/伸長のためのハードウェア/ソフトウェアを簡単に開発できます．携帯用MP3プレーヤやパソコン上で動作するMP3エンコーダ/デコーダ・ソフトが次々に開発され，きわめて安価に提供されてきました．

専門委員会
ISO/IEC JTC1/SC2/WG8の中の非公式の分科会として発足したもので，JTCはJoint Technical Committee，SCはSubCommittee，WGはWorking Groupの略．なお，WG8はその後SC2から独立して，現在はSC29/WG11となっている．このSC29は画像圧縮を専門に扱う副委員会であり，静止画圧縮（JPEG）のWG1，動画圧縮（MPEG）のWG11，，マルチメディア/ハイパーメディア圧縮（MHEG）のWG12の三つの作業部会から構成される．

mpman
MP3再生用の携帯音楽プレーヤは，ほかのメーカでも早くから開発していたが，著作権問題が解決できないため発売を見合わせていた．とくに，従来CDプレーヤなどを発売していた大手電機メーカの多くはレコード会社を系列にもっており，MP3市場に参入できなかった．mpmanに続いて韓国，ドイツ，アメリカなどの企業が携帯MP3プレーヤを発売したが，いずれも音楽業界と関連の薄いメーカだった．

〈表7-2〉MPEGオーディオの規格

	Layer1	Layer2 (MP2)	Layer3 (MP3)	AAC (Advanced Audio Coding)
MPEG-1	○	○	○	−
MPEG-2	○(拡張あり)	○(拡張あり)	○(拡張あり)	○

←より簡単　　より複雑→

レイヤが高くなるほど処理は複雑になるが，そのかわり同じ音質でより高圧縮化できる．同じ圧縮率ならより高音質が得られる．
AACはMPEG-2から採用された．MP3とは互換性がないが，同じ音質でより高圧縮化できる．最近は携帯用プレーヤにも採用されている．
MPEG-2でのLayer1～3のおもな拡張部分は，低サンプル・レート・モードとマルチモード・ステレオ・モードの二つが追加されたこと．それ以外はMPEG-1のLayer1～3とほぼ同じ．低サンプル・レート・モードは，標準の1/2 (16kHz，22.05kHz，24kHz) のPCM入力をサポートするもの．マルチモード・ステレオは，たとえばリア・チャネルを追加したサラウンド・システムに対応できる．

これらのことからMP3は急速に普及しましたが，いくつかの問題も生じています．

まず，MP3が市販の音楽データの違法コピーに利用され，音楽著作権を侵害する恐れがあることは早くから指摘されており，音楽関係のメーカ，団体などは違法コピーの防止に力を入れてきました．また，家電メーカやコンピュータ・メーカとの間で**著作権保護や使用料の徴収などの協議**を進めてきましたが，ルールが確立しないうちにMP3は急速に普及していきました．

このため，大手の家電メーカやコンピュータ・メーカではMP3関連製品の商品化が遅れ，MP3市場は中小メーカや個人が主体となって自由な発展をとげてきました．それはそれでいいのですが，オリジナルのMP3フォーマットに対して何らかの情報を付加したさまざまな**拡張MP3フォーマット**が普及するという問題が生じています．MP3そのものはISO/IECによって承認された国際規格ですが，これらの拡張フォーマットは規格の正式な拡張ではなく，拡張フォーマットどうしの互換性も保証されていません．

また，それらとは別の問題として，画像フォーマットのLZW方式の特許ライセンス問題と同様に，MP3の圧縮/伸長技術にも特許ライセンスの問題が生じています．

これらの問題については，MP3の仕様の解説の後にまとめて述べたいと思います．

著作権保護や使用料の徴収などの協議
それまでにあったDATやMDなどディジタル録音可能な機器は，いずれもコピー回数を制限する機構を備えたり，機器メーカから著作権団体に使用料を代行して支払うなどのルールが作られていた．その使用料は，機器やメディア（テープ，ディスク）の価格に含まれている．

拡張MP3フォーマット
MP3は音声の圧縮方式としては優れているが，もともと音楽データを交換するためのフォーマットではないため，タイトルやアーティストなどの付加情報を記録する場所がない．そのため，自発的にさまざまな拡張フォーマットが提案され，普及してきた．

MPEG-1のシステム仕様とMP3 (MPEG-1 Audio Layer3) オーディオ仕様

MPEG-1のシステム仕様は，**VHSビデオ相当の画質**の動画データを，1.5Mビット/s以下のビット・レートでリアルタイム再生することを目標としています．この1.5Mビット/sは画像と音声を合わせたもので，音声だけのビット・レートは標準で128kビット/sです（**図7-1**）．

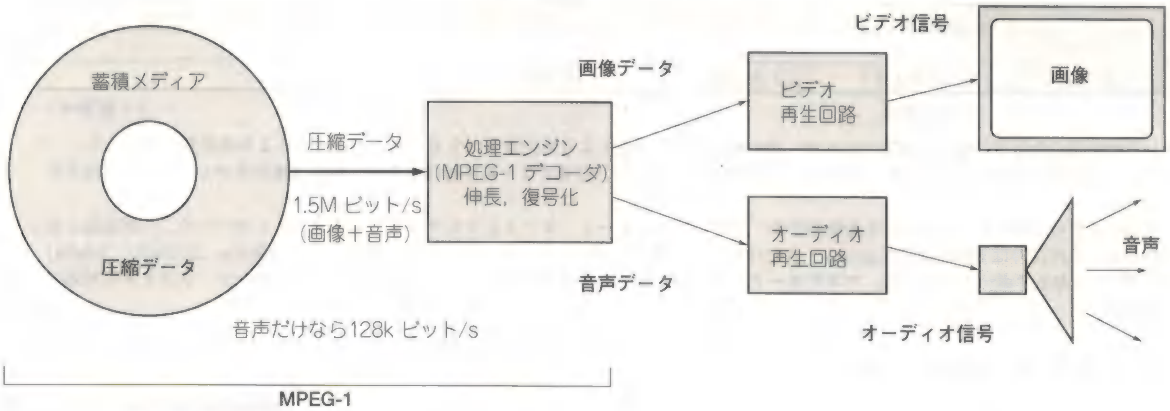
ビット・レートとは，動画や音声の1秒あたりのデータ・サイズを示すものです．たとえば，1.5Mビット/sの動画データは1分間で90Mビット＝11.25Mバイト，128kビット/sの音声データは1分間で7.68Mビット＝960kバイトとなります．MPEG-1動画を再生するためには，圧縮データを記録したメディア（CD-ROM，HDD，Webサーバなど）から少なくとも1.5Mビット/sのレートでデータを読み出し，処理エンジン（MPEGデコーダLSIまたはCPU）に転送しなければなりません．そのため，この値は**転送レート**とも呼ばれます．

元データが同じであれば，圧縮率を高くすればするほどビット・レートが下がり，記録メディアや転送部分の負荷は軽くなります．ただし，処理エンジンは元

VHSビデオ相当の画質
デコードされたMPEG画像は，パソコン画面やTV画面などさまざまな環境で再生される．そのため，再生画質を定量的に表現するのは難しく，VHSビデオ相当というあいまいな表現に落ち着いた．

転送レート
MPEG-1やMPEG-2ではちょっと無理だが，MPEG-4のようにインターネットからダウンロードしながらリアルタイムで再生（ストリーミング再生）することを考えると，ビット・レートはまさにダウンロードの転送レートそのものだ．

〈図 7-1〉MPEG-1 のシステム仕様



データを復元するのですから，圧縮データのビット・レートが低くても負荷はまったく軽くなりません．逆に，伸長，復号化の処理が複雑になるため，処理エンジンの負荷は重くなる場合があります．処理エンジンの負荷を減らすために処理を単純にすれば，画質や音質の犠牲が大きくなります．

一般的なインターネット環境の転送速度は 1.5 M ビット/s よりずっと低速なので，Web サーバ上の MPEG-1 データをダウンロードしながら同時に再生すること（ストリーミング再生）は困難と考えられます．MPEG-1 は，CD-ROM や HDD など大容量の蓄積メディアを利用することを想定しています．たとえば，ビデオ CD の規格に MPEG-1 が採用されています．

ただし，音声だけなら 128 k ビット/s ですから，一般の ISDN の 2 チャネルで**ぎりぎりストリーミング再生できる**可能性もあります．さらに，MPEG-1 Audio では必要に応じて音声データのビット・レートを変更できます．Layer1 では 32 k ～ 448 k ビット/s, Layer2 では 32 k ～ 384 k ビット/s, Layer3 では 32 k ～ 320 k ビット/s の範囲でビット・レートを選択できます．上限についてはさらに高くすることも可能です．

ぎりぎりストリーミング再生できる
回線の混雑などにより，ちょっとでも伝送速度が低下すると，データが間に合わなくなって音がとぎれたりする．

CD-DA（音楽 CD）の音声データは，サンプリング・レート 44.1 kHz で 1 サンプルあたり 16 ビット×2 チャネルですから，ビット・レートは 44100×16×2＝1411200 ビット/s です．これを 128×1000＝128000 ビット/s に圧縮すれば，圧縮率は 1/11.025 になります．

同じ 128 k ビット/s のビット・レートでくらべれば，Layer1 は**もっとも処理が単純で**処理エンジンの負荷が軽いかわりに，音質はもっとも低下します．逆に，Layer3 は**もっとも処理が複雑で**処理エンジンの負荷が重いかわりに，音質の低下は最小にできます．一般に「CD-DA 並みの音質を保ちながらデータを約 1/11 に圧縮できる」と言われるのは Layer3 のことです．

もっとも処理が単純で
元データの情報を間引く方法が単純なので，同じ量だけ情報を間引いても聴感上目立ってしまう．

もっとも処理が複雑で
複雑な方法を使って目立たないように元データの情報を間引くので，同じ量だけ情報を間引いても音質はあまり低下しない（ように聞こえる）．

MP3 は MPEG-1 Audio Layer3 の略であり，Layer1 ～ 2 は MP3 と直接関係がありません．以下では，Layer3 だけについて述べます．

MP3 のシステム仕様

MP3 エンコーダは，CD，DAT などの高音質 PCM 音声データを入力データとして圧縮，符号化を行います．圧縮方式は基本的には 1 種類だけです．非圧縮モードは存在しません．

入力データのサンプリング・レートは48kHz(DATなど)，44.1kHz(CDなど)，32kHzの3種類あります．ビット数は16ビットのみです．チャネル数は1または2で，モノラル（1チャネル），ステレオ（2チャネル），独立2チャネルを扱うことができます．

またステレオ方式として，通常の2チャネル・ステレオのほかに**ジョイント・ステレオ**をサポートします．ジョイント・ステレオにはIntensity方式とM/S方式があります．

Intensity方式は，モノラル信号（L＋R）をベースにして，それにL，R各チャネルのスケール・ファクタ（和信号に対する比）を加えて左右の方向感を再現するものです．2チャネル・ステレオにくらべてデータ・サイズを削減できます．ただし，カットすると聴感上の影響が大きい低域成分は2チャネル信号をそのまま圧縮します．

M/S（Middle/Sides）方式は，各チャネルの和信号（L＋R）と差信号（L－R）を用いてステレオ信号を表すものです．両チャネルに共通の音成分（音場の中央付近の音）は差信号から除かれるため，差信号は和信号より0に近い値が多くなり，データ・サイズを削減することが可能です．

MP3ではこのIntensity方式とM/S方式を組み合わせて利用します．MP3エンコーダは元データの特性からIntensity方式とM/S方式のどちらがデータ・サイズを小さくなるかを判定し，自動的に**方式を切り替えながら**元データを圧縮，符号化します．

出力データのビット・レートは128kビット/sが標準で，32k～320kビットの範囲から選択することもできます．MPEG-2ではさらに高圧縮のモード（32kビット/sより低いビット・レート）もサポートしますが，MPEG-1では32kビット/sが下限です．

なお，MPEG-1の規格書ではエンコーダの圧縮，符号化アルゴリズムは**具体的に規定されていません**．付録の中に参考として例があげられているだけです（Annex C: Encoding Process）．MPEGの規格で規定されているのは，圧縮データのフォーマットと，デコーダの伸長，複合化アルゴリズムだけです．

また，MPEG-1 Audioでは音声データ・ストリームのフォーマットが規定されているだけで，音声データ・ファイルのフォーマットは規定されていません．画像データと音声データをまとめて一つの動画データ・ファイルにする方法は，MPEG-1の規格書のPart1（システム）で規定されています．

一般にMP3ファイルと呼ばれているものは，MPEG-1 Audioで規定された音声データ・ストリームをそのままファイル化したものです．ただし，あとで述べるようにそれに拡張情報を付加した変種のMP3ファイルも広く流通しています．

DAT
Digital Audio Tapeの略．民生用のディジタル録音機器としては最初のもので，CDを上回る音質の仕様をもち，業務用としても使われている．単に音楽を楽しむにはちょっとオーバ・スペックだった．後に，より手軽に使える民生用のディジタル録音機器として，MD（MiniDisc）とDCC（Digital Compact Cassette）が登場し，その中でMDが広く普及した．

ジョイント・ステレオ
MP3ではIntensity方式とM/S方式の二つがあるが，Layer1～2ではIntensity方式だけ．

方式を切り替えながら
どちらか一方だけを使うこともできる．

具体的に規定されていません
MP3ではエンコーダによって，処理のスピード，圧縮の効率，再生したときの音質は大きく変わる．符号化の具体的なアルゴリズムは開発者に任されているが，開発はなかなか難しく，ほとんどのエンコーダ・ソフトは市販（またはフリー）の処理エンジンを利用している．代表的なエンコード・エンジンには，Xing社（速度に定評がある），Fraunhofer社（音質に定評がある）と，ISOのサンプル・ソースをもとにしたもの（Lame，Bladeencなど）がある．

圧縮技法の概要

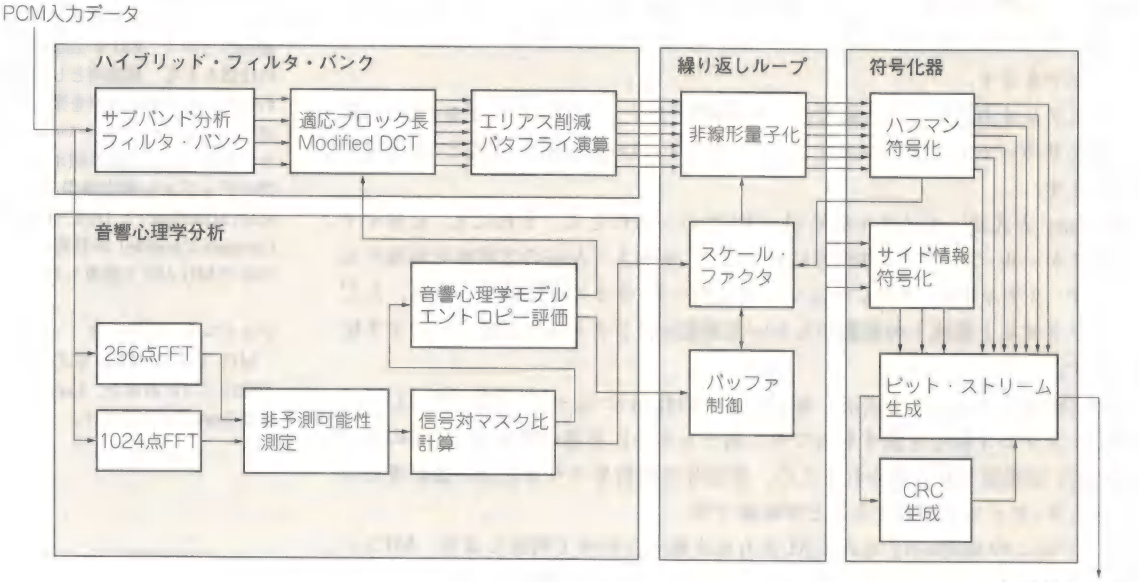
MP3では音声データ圧縮で使われるさまざまな技法を複雑に組み合わせて，音質の低下を防ぎながら約1/11という高い圧縮率を実現しています（**図7-2**）．

まず，元のPCMデータはフィルタ・バンクを用いて**32個のサブバンドに分割**され，さらに変形離散コサイン変換（Modified Discrete Cosine Transform）を用いて周波数分解能は実質的に18倍に拡張されます．これにより，MP3では32×18＝576個のスペクトルに対応する576サンプルのデータが生成されます．さらに，エリアスを除去するためのバタフライ演算が行われます．

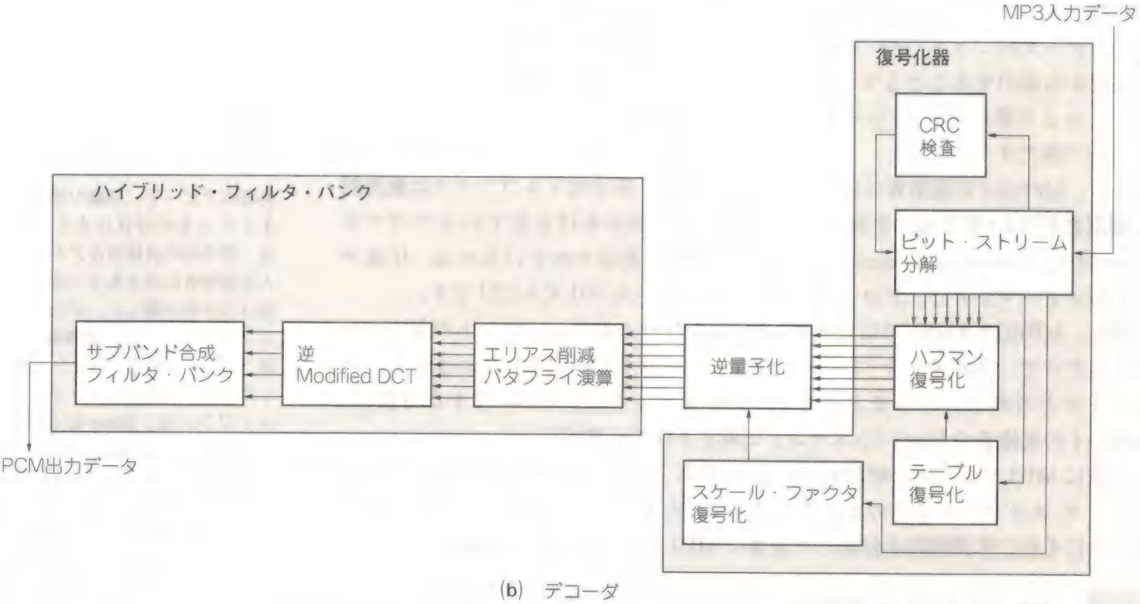
MP3ではこの576サンプルを1単位のデータとして扱い，granule（粒子）と

32個のサブバンドに分割
音声帯域を32個のサブバンドに分割するのはLayer1～3に共通の方式．Layer1～2ではその32個のまま量子化，スケール・ファクタ演算を行うが，Layer3ではMDCTを用いてさらに周波数分解能を拡張している．

〈図 7-2〉 MP3 のブロック図



(a) エンコーダ



(b) デコーダ

量子化
要するにデータの端数を切り捨てるということ．JPEG，MPEG（ビデオ，オーディオ）などの不可逆圧縮はこの量子化のステップを含む．一方，可逆圧縮では量子化を行わない．

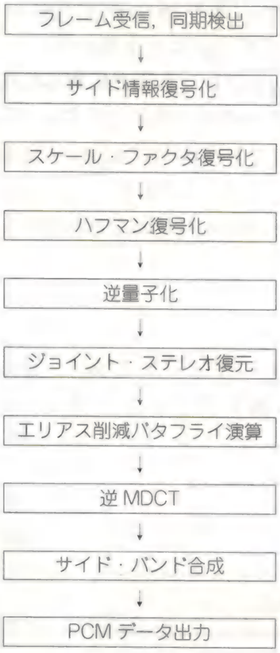
ハフマン符号化
量子化されたデータをさらに圧縮する．この圧縮は可逆圧縮．

呼びます．さらに，granule の 2 個分すなわち 1152 サンプルのデータが 1 フレームを構成します．

一方，音響心理学モデルに基づき，各スペクトルごとに可聴ノイズ限界と信号/マスク比が計算され，各サンプル値のビット割り当てとスケール・ファクタが決定されます．これを用いて聴感上影響の少ないサンプル値を抑圧し，非線形に**量子化**することによって，データ・サイズは大幅に削減されます．非線型量子化されたデータは**ハフマン符号化**され，さらに圧縮されます．

このスケール・ファクタとハフマン符号化データが圧縮データの本体になります．また，ビット割り当て情報やハフマン・テーブル選択情報など，圧縮データ

＜図 7-3 ＞ MP3 デコーダの復号化手順



の伸長，復号化に直接必要となるパラメータはサイド情報としてデータに付加されます．

MP3 デコーダは，MP3 ファイルから 1 フレームごとにサイド情報を取り出し，それを用いてハフマン復号化とスケール・ファクタの復号化を行い，逆量子化によって各スペクトルのサンプル値を復元します．さらに，逆 MDCT とフィルタ・バンクを用いて PCM データを復元し，音声を再生できます（**図 7-3**）．

MP3 データのフォーマット

MPEG-1 の規格書では，C の構文を用いて圧縮した音声データ・ストリームのフォーマットを定義しています．ここでは，なるべく図を加えて解説を行います．

● オーディオ・シーケンスとフレーム

オーディオ・シーケンスはフレームの繰り返しでできています．あとで述べるように，各フレームの先頭は 12 ビットの同期情報（syncword）です．フレームの後に syncword が続くなら，それは次のフレーム開始と見なすことができます．フレームの後に syncword が検出されなくなれば，オーディオ・シーケンスの終了と見なすことができます．MP3 ファイルはこの 1 個のオーディオ・シーケンスをそのままファイル化したものと見なせます（**図 7-4**）．

もともとの MPEG-1 の規格では画像データと音声データを同期させながら並列に処理する必要があります．そのため，画像と音声をそれぞれ**一定長のパケット**に分割し，それらを交互に転送することによって時分割の並列処理を行います．MPEG-1 デコーダは受信したパケットを画像と音声に分離し，それぞれのデータ・ストリームを復元します（**図 7-5**）．

MP3 の場合は画像なしの音声だけを扱うので，**図 7-4** のように音声フレーム

一定長のパケット
可変長のフレームをつないで一連のシーケンスを作り，それを一定長ずつ切り出してパケットを作る．

だけが連続して転送されます．前述のように，各フレームは1152サンプルのデータを含み，576サンプルずつ2個のgranuleに分けられます．

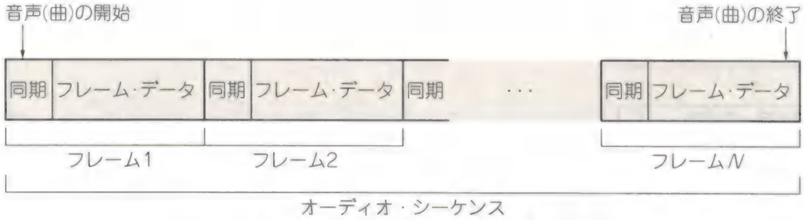
同じ1152サンプル分といっても，ビット割り当てやスケール・ファクタの違いにより，圧縮データのサイズは**フレームごとに異なります**．そこで，圧縮データのサイズが大きい部分ではデータの一部を先行するフレームに収容し，圧縮データのサイズが小さい部分ではフレームの空き部分に後続のフレームのデータを収容するという方式をとっています．MP3デコーダでは，先行して送られてきた圧縮データを保持しておき，フレームが転送されるのを待って伸長，復号化を行います(**図7-6**)．

フレームごとに異なります
フレームは可変長だが，均一の長さにすることが望ましい．

● フレームの構成

フレームは，ヘッダ，エラー・チェック(CRC)，音声データ，補助データの四つの部分から構成されます(**図7-7**)．エラー・チェックと補助データはオプショ

〈図7-4〉オーディオ・シーケンス

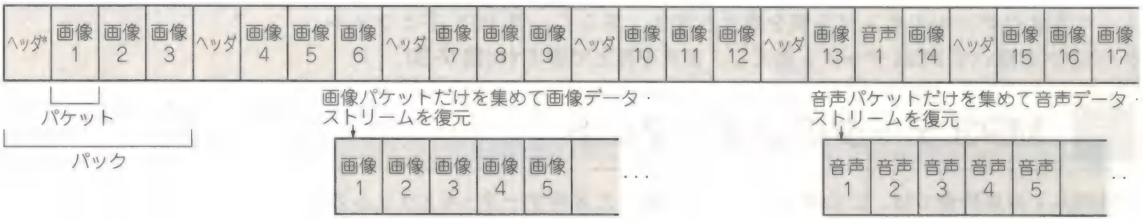


(a) オーディオ・シーケンスの構成

```
audio sequence( )
{
   while(nextbits==syncword) {
      frame( )
   }
}
```

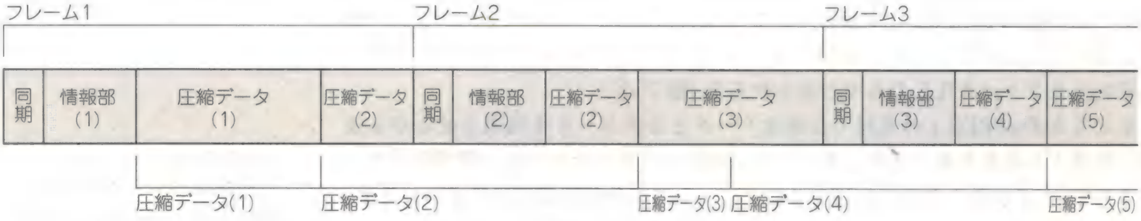
(b) 書式

〈図7-5〉MPEG-1動画データ・ストリームの例



画像データ，音声データは，それぞれ一定長のパケットに分割され，時分割で多重化される．
音声にくらべて画像は圧倒的にデータ量が多い(ビット・レートが高い)ので，その比に合わせてパケット比が決定される．この例では13：1．
数個のパケットを集めたパックを一つの単位として扱う．パックにはそれぞれヘッダが付加される．
先頭のパックのヘッダ(図のヘッダ*)だけは，システム・ヘッダと呼ばれる特別のヘッダを含む．

〈図7-6〉MP3のフレームと圧縮データの配置



圧縮データはそのフレームの時間内にデコード処理を行わなければならないので，後ろ方向にはみ出すことは許されない．
フレームNの圧縮データのサイズが大きいときは，それより先行するフレーム(フレームN-1，フレームN-2，…)の空きにつめて転送する．

ンで，省略される場合もあります．

ヘッダには**一連のオーディオ・シーケンスに関する情報**が含まれています．ヘッダは各フレームごとに付属していますが，基本的にはすべて同じ内容です．もし必要なら，シーケンスの途中でヘッダのパラメータを変更することも可能です．たとえば，曲の進行に合わせてビット・レートを変更する可変ビット・レートが使われています．

一連のオーディオ・シーケンスに関する情報
音声を再生するのに必要な，元データのサンプリング・レートやチャネル数，どんな方法で圧縮したかなどの情報からなる．

また，ヘッダにはフレームの同期情報（syncword）が含まれており，画像と同期をとりながら一定レートで音声を再生できるようになっています．

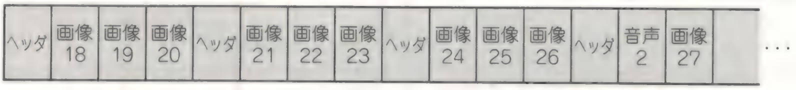
音声データには，そのフレームを再生するのに直接必要なサイド情報と，圧縮データ本体が含まれています．MP3デコーダは，サイド情報を使って圧縮データを伸長，復号化します．前述のように，圧縮データの一部はフレームよりも先行して転送されている場合があります．

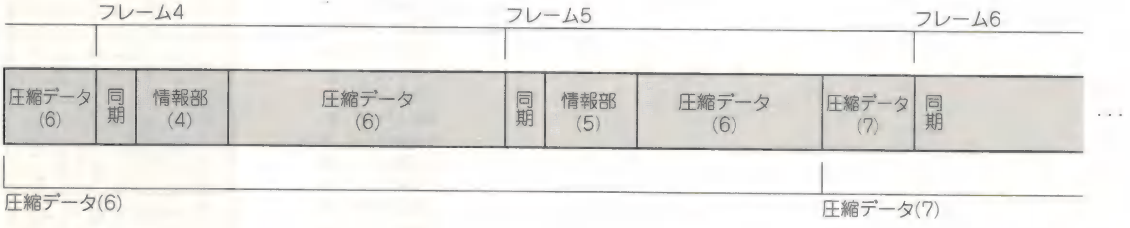
● **ヘッダの構成**

ヘッダは32ビット固定長で，その最初の12ビットがsyncword（12ビットの'1'の連続）です．フレームの先頭はかならずsyncwordで始まり，同じフレーム内に複数のsyncwordが含まれることはありません．syncwordを用いて再生の同期をとることができます．また，MP3ファイルの先頭にはかならずこのsyncwordが現れますから，ファイルがMP3フォーマットであることの確認にも利用できます．

ヘッダの残り20ビットはそのオーディオ・シーケンスに関するさまざまな情報をまとめたものです（**図7-8**）．

IDはMPEGの種類を示す1ビットの情報で，ID＝0はMPEG-2，ID＝1は





MPEG-1 です．

Layer は 2 ビットで，Layer＝11 は Layer1，Layer＝10 は Layer2，Layer＝01 は Layer3 を示します．2 進数値がレイヤ番号を示すというわけではないので，注意が必要です．

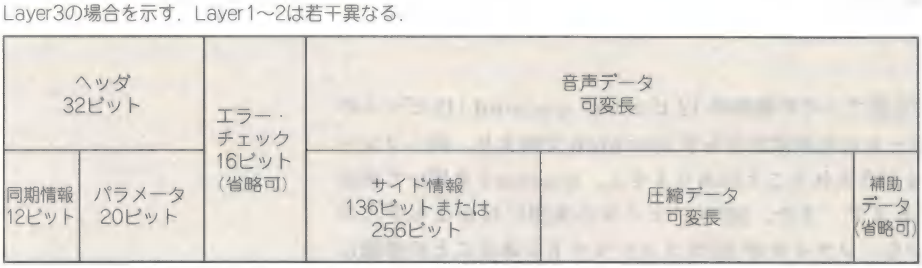
protection_bit

0 がエラー・チェックあり，1 がエラー・チェックなしというのは，一般的な感覚とは逆なので注意が必要．

protection_bit はエラー・チェックが有効かどうかを示すビットです．protection_bit＝0 はエラー・チェックあり，protection_bit＝1 はエラー・チェックなしです．

bitrate_index はビット・レートを示す 4 ビットの情報です．bitrate_index の値が同じでも，それが**実際に示すビット・レートは Layer によって異なる**ので注意

〈図 7-7〉 フレーム



（a） フレームの構成

```
frame( )
{
    header( )
    error_check( )
    audio_data( )
    ancillary_data( )
}
```

（b） 書式

〈図 7-8〉 ヘッダ

（a） ヘッダの構成とパラメータ

が必要です．Layer3の場合，bitrate_index＝1001が標準の128kビット/sを表します

sampling_frequencyは元データのサンプリング・レートです．sampling_frequency＝00は44.1kHz，sampling_frequency＝01は48kHz，sampling_frequency＝10は32kHzを示します．

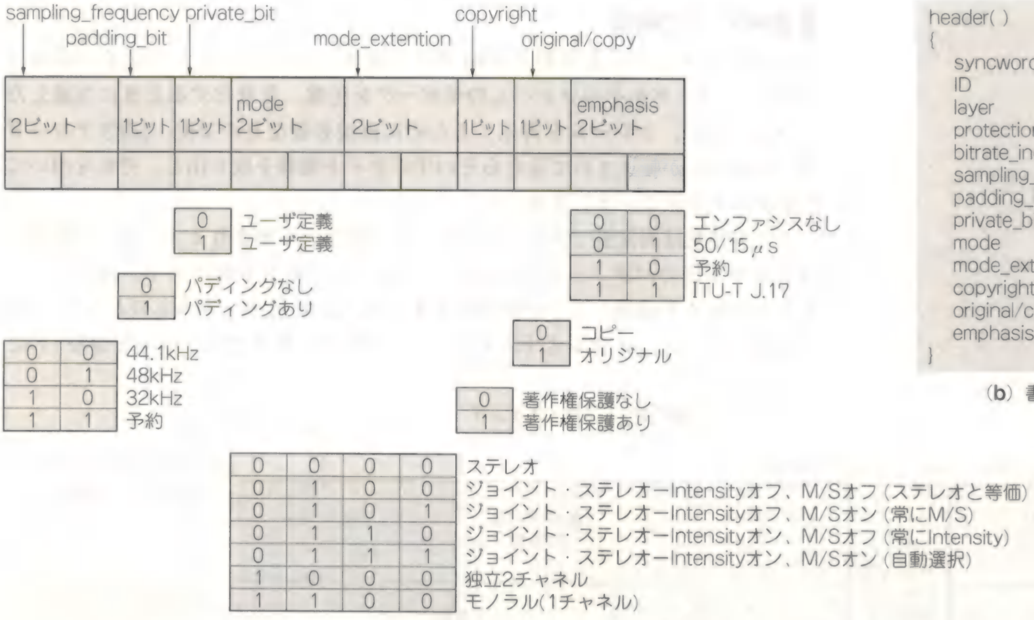
padding_bitはフレーム長を調整するための余分なスロット（バイト）が付加（パディング）されているかどうかを示します．padding_bit＝0はパディングなし，padding_bit＝1はパディングありです．サンプリング・レート44.1kHzではかならずパディングを行い，フレーム長を均一にします．他のサンプリング・レートではパディングは必須ではありませんが，なるべく行うように推奨されています．

private_bitはユーザが任意に定義できる1ビットの情報です．現在のISO/IEC規格ではこのビットを使っていませんし，将来にわたっても使わないことを保証しています．

mode（2ビット）とmode_extention（2ビット）はステレオ・モードを示します．mode＝00は通常の2チャネル・ステレオ，mode＝01はジョイント・ステレオ（IntensityおよびM/S），mode＝10は独立2チャネル（日本語チャネルと英語チャネルのような使い方ができる），mode＝11はモノラル（1チャネル）です．また，mode＝01（ジョイント・ステレオ）のとき，mode_extensionによってIntensity方式とM/S方式のどちらが使われるか（もしくは両方）を示します．

実際に示すビット・レートはLayerによって異なる

bitrate＝0000がフリー・フォーマット，0001が32kビット/sというのは各レイヤに共通．しかし，0010はLayer1では64kビット/s，Layer2では48kビット/s，Layer3では40kビット/sというようにそれ以降はLayerによって割り当てが変わる．




```
header( )
{
  syncword
  ID
  layer
  protection_bit
  bitrate_index
  sampling_frequency
  padding_bit
  private_bit
  mode
  mode_extention
  copyright
  original/copy
  emphasis
}
```

（b）書式

違法コピー防止などの著作権保護に利用できます
ただし，MP3は一般のパソコンで扱えるファイルなので，ユーザがこの情報を書き換えることも容易にできる．実際には使われていないようだ．

copyrightとoriginal/copyは**違法コピー防止などの著作権保護に利用できます**．copyright＝1はこの音声データは著作権保護が必要であることを示し，original/copy＝1はこれが複製でないオリジナル・データであることを示します．

emphasisはノイズ・リダクションなどの目的で高域に加えられる強調（エンファシス）の種類を示します．emphasis＝00はエンファシスなし，emphasis＝01は50/15 μs, emphasis＝10は予約，emphasis＝11はITU-T J. 17準拠のエンファシスを示します．

● エラー・チェック

CRC-16方式
CRC（Cyclic Redundancy Check）は原理は同じだが，使用する計算式によっていくつかの種類に分かれている．

エラー・チェックは**CRC-16方式**による2バイトのCRCコードです．この機能はオプションなので，省略することもできます．ヘッダのprotection_bitが0のときヘッダと音声データの間にCRCが挿入されます．protection_bitが1のときエラー・チェックなしです．

このCRC計算の対象となるのは，ヘッダの後半部分と，音声データのサイド情報部分です（**図7-9**）．32ビットのヘッダのうち，syncword（12ビット），ID（1ビット），Layer（2ビット），protection_bit（1ビット）の16ビットはCRC計算から除外され，bitrate_index以降の16ビット（ビット16～ビット31）がCRC計算の対象となります．

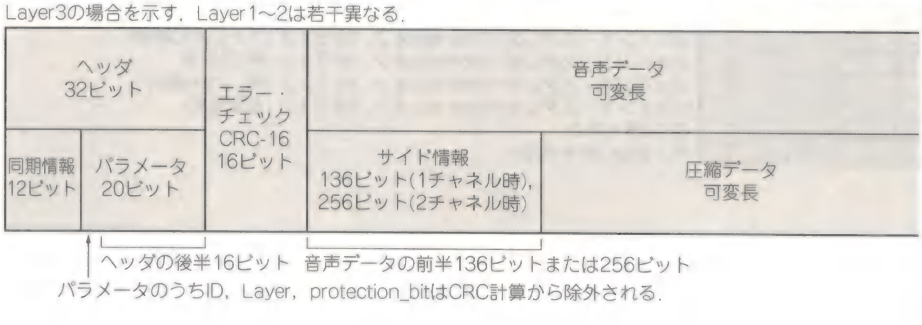
また，サイド情報は固定長で，モードがモノラル（1チャネル）のとき17バイト（音声データのうちビット0～ビット135），ステレオ（ジョイント・ステレオ，独立2チャネルを含む）のとき32バイト（音声データのうちビット0～ビット255）です．

● 音声データの構成

音声データはサイド情報と圧縮データ本体に大別できます．サイド情報はMP3エンコーダがそのフレームの元データを圧縮，符号化するときに生成したパラメータで，フレームを再生するために直接必要なものです．MP3デコーダは，フレームが転送されてきたらその中のサイド情報を取り出し，それを用いて圧縮データをデコードします．

サイド情報は固定長であり，音声データの先頭から始まります．サイド情報にはチャネル別のパラメータも含まれているため，モードが1チャネルか2チャネルかでサイド情報の長さが変わります．モノラル（1チャネル）のとき17バイト（136ビット），ステレオ（ジョイントステレオ，独立2チャネルを含む）のと

〈図7-9〉エラー・チェック



(a) CRCの計算範囲

$G(X)=X^{16}+X^{15}+X^2+1$

(b) CRCの計算式

```
frame( )
{
   header( )
   error_check( )
   audio_data( )
   ancillary_data( )
}
```

(c) 書式

き32バイト（256ビット）です．サイド情報にはさまざまなパラメータが含まれているので，おもなものだけを説明します（図7-10，p.116）．

サイド情報の先頭はmain_data_beginと呼ばれる9ビットの値で，**圧縮データ本体の開始位置のオフセット**を示します．main_data_begin＝0ならば，圧縮データ本体はそのフレームのサイド情報の直後から始まります．main_data_begin＞0ならば，その値の分だけ先行フレーム側にアドレスをさかのぼった位置から始まります．ただし，ヘッダ，エラー・チェック，サイド情報はアドレス計算から除外します．

なお，圧縮データ本体の開始位置が，そのフレームのサイド情報の直後（オフセット＝0）より**後にくることは許されません**．そのため，main_data_beginは符号なし整数として扱われ，負になることはありません．

part2_3_length[gr][ch]は，グラニュールごと，チャネルごとの圧縮データ本体の長さ（バイト数）を示す2次元配列です．1個のフレーム（1152サンプル）は2個のグラニュール（各576サンプル）から構成されるので，grの値は0，1のいずれかです．また，モードがモノラルならchの値は0だけ，ステレオならchの値は0, 1のいずれかです．

main_data_beginは最初のグラニュール（gr＝0）の圧縮データ本体の開始位置を示します．part2_3_length[gr][ch]を用いれば，次のグラニュール（gr＝1）の圧縮データ本体の開始位置や，その直後に置かれる補助データの開始位置を計算できます．

圧縮データ本体の開始位置のオフセット
MP3では，そのフレームの圧縮データを，それより前のフレームの空きスペースにつめて先行して送ることができる．そのため，圧縮データがどこから始まるかの情報をもたせる必要がある．

後にくることは許されません
圧縮データをそのフレームより先行して送ることはできるが，遅らせることはできないということ．

MP3における著作権保護の問題

もともと，CD-DAのPCM音声データは簡単にパソコンに取り込むことができ，**WAVEやAIFF**などのフォーマットでファイル化できます．さらに，それをMP3フォーマットに変換することも容易であり，市販のCD-DAの楽曲を圧縮したMP3ファイルを誰でも簡単に作ることができます．これらのファイルは個人で私的に利用するには何も問題ありませんが，それを他者に配付するのは著作権法に違反します．

この違法コピーの問題はWAVEやAIFFでも同じであり，MP3だけが問題になっているわけではありません．しかし，WAVEやAIFFのファイルはデータ・サイズがきわめて大きく，保存や配布には不向きです．**MP3ファイルはきわめてコンパクトで**音楽データの配布に適しているために，とくにMP3が問題視されているわけです．

MP3の規格自体は違法コピーを防止するための最低限のしくみをもちます．MP3ファイルのヘッダ部分には，copyrightフラグとoriginal/copyフラグという二つのフラグがあります．copyright＝1は著作権保護が必要なデータであることを示し，original/copy＝0はこのデータが何かのソースからコピーされたものであることを示します．これを利用して音楽データのコピーや配布を制限することは可能です．

しかし，MP3ファイルはパソコン上の汎用データなので，ヘッダ情報は簡単に書き換え可能です．この二つのフラグの利用をルール化したとしても，実際に違法コピーを防止する効果がどれほどあるかは疑問視されています．

このため，MP3が普及のきざしを見せ始めた1998年初頭から，**音楽著作権を管理する団体**（日本ではJASRAC）やレコード会社，アーティストなどは，音楽データの違法コピーや頒布に対して警告を発してきました．1998年末には

WAVEやAIFF
WAVEはWindows標準の音声データ・フォーマット，AIFFはMacintosh標準の音声データ・フォーマット．いずれもPCM音声そのままで，圧縮は行わない．

MP3ファイルはきわめてコンパクトで
MP3と同程度またはそれ以上の音質で，データをさらにコンパクトにできるという新しいフォーマットも作られている．MP3の場合，CD並みの音質を保って約1/11に圧縮できるというのがうたい文句だが，AAC（Advanced Audio Coding），TwinVQ（Transform-domain Weighted Interleave Vector Quantization），WMA（Windows Media Audio）などのフォーマットは同程度の音質で最大1/20ぐらいまでの圧縮ができると言われている．

〈図 7-10〉音声データの構成

Layer3の場合を示す．Layer 1～2は若干異なる．

main_data_begin	private_bits	ch=0,1，scfsi_band=0,1,2,3 scfsi[ch][scfsi_band]	gr=0，ch=0					
			part2_3_length [gr][ch]	big_values [gr][ch]	global_gain [gr][ch]	scalefac_compress [gr][ch]	window_switching_flag [gr][ch]	window_switching_flag=1のとき: block_type[gr][ch] 2ビット window_switching_flag=0のとき: region=0,1,2 table_select[gr][ch]
圧縮データの開始位置	ユーザ定義	スケール・ファクタ選択	圧縮データのデータ長					
9ビット	3ビット	1ビット×2×4配列	12ビット	9ビット	8ビット	4ビット	1ビット	5ビット×3配列

グラニュール，チャネルごとのサイド情報部（12+9+8+4+1+2+1+5×2+3×3+1+1+1=59ビット）

サイド情報（9+3+1×2×4+59×4）=256ビット

(a) 2チャネル時（ch=0,1）

main_data_begin	private_bits	scfsi_band=0,1,2,3 scfsi[ch][scfsi_band]	gr=0，ch=0					
			part2_3_length [gr][ch]	big_values [gr][ch]	global_gain [gr][ch]	scalefac_compress [gr][ch]	window_switching_flag [gr][ch]	window_switching_flag=1のとき: block_type[gr][ch] 2ビット window_switching_flag=0のとき: region=0,1,2 table_select[gr][ch]
圧縮データの開始位置	ユーザ定義	スケール・ファクタ選択	圧縮データのデータ長					
9ビット	5ビット	1ビット×4配列	12ビット	9ビット	8ビット	4ビット	1ビット	5ビット×3配列

グラニュールごとのサイド情報部（12+9+8+4+1+2+1+5×2+3×3+1+1+1=59ビット）

サイド情報（9+5+1×4+59×2）=136ビット

(b) 1チャネル時（ch=0）

音楽著作権を管理する団体

公的な場で音楽を演奏したり録音ソースを再生することや，楽譜や録音ソースを複製して交換や頒布することなどは，禁じられていたり，著作権者に対して相応の使用料を支払うことが要求される．だからといって，著作権者個人がそれらの活動を監視して料金を徴収して回るのは不可能なので，それを代行してくれる団体が組織されている．

RIAA（米国レコード工業会）が中心となって音楽業界やコンピュータ業界の大手メーカを集めた業界団体，SDMI（Secure Digital Music Initiative）を組織し，著作権保護や使用料徴収のしくみを備えたディジタル音楽配信技術のルール作りを進めてきました．

SDMIでは1999年7月に著作権保護を備えた携帯用プレーヤの仕様としてSDMI Portable Device Specification Part1を発表しました．それを契機として，それまでMP3プレーヤ市場への参入を避けてきた大手家電メーカから，SDMI仕様に準拠した携帯用プレーヤが商品化されるようになりました．

SDMIでは，フェーズ1，フェーズ2という2段階のステップで著作権保護技術を普及させていくことを表明しています．まずフェーズ1では，MP3などと共存しながらSDMI仕様の機器や音楽データを普及させていきます．SDMI仕様がある程度普及したら，フェーズ2として著作権保護をもたない音楽データの流通の抑制をはかります．現在はまだフェーズ1の段階であり，SDMI準拠の携帯用プレーヤの中には著作権保護をもたない既存の音楽データを再生可能なもの

			perflag [gr][ch]	scalefac_scale [gr][ch]	count1table_select [gr][ch]	gr=0, ch=1	gr=1, ch=0	gr=1, ch=1	main_data( ) 圧縮データ本体
mixed_block_flag [gr][ch]	region=0,1 table_select[gr][ch][region]	window=0,1, 2 subblock_gain[gr][ch][window]				繰り返し	繰り返し	繰り返し	
1ビット	5ビット×2配列	3ビット×3配列							
region0_count [gr][ch]	region1_count[gr][ch]								
4ビット	3ビット		1ビット	1ビット	1ビット	59ビット	59ビット	59ビット	可変長

			perflag [gr][ch]	scalefac_scale [gr][ch]	count1table_select [gr][ch]	gr=0, ch=1	main_data( ) 圧縮データ本体
mixed_block_flag [gr][ch]	region=0,1 table_select[gr][ch][region]	window=0,1,2 subblock_gain[gr][ch][window]				繰り返し	
1ビット	5ビット×2配列	3ビット×3配列					
region0_count [gr][ch]	region1_count[gr][ch]						
4ビット	3ビット		1ビット	1ビット	1ビット	59ビット	可変長

も含まれています．

SDMI仕様は音楽データや携帯用プレーヤおよびその**記録メディアが備えるべきの著作権保護の要件**を規定していますが，具体的な実現方法までは規定していません．各メーカではSDMIに準拠した音声フォーマットや著作権保護技術をそれぞれに提唱しています．たとえば，Liquid Audio社の提唱するSP3，NTTと神戸製鋼が提唱するInfoBind，ソニーが提唱するMagic GateおよびOpenMGなどがあります．既存のMP3ファイルに電子透かしなどの著作権保護技術を取り入れる方法もいくつか提唱されています．

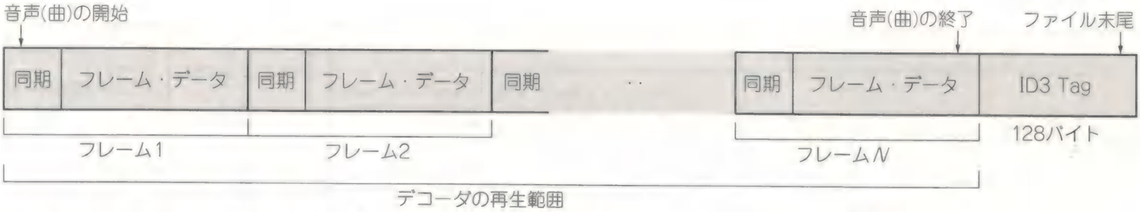
記録メディアが備えるべき著作権保護の要件

スマートメディア，メモリースティック，マルチメディアカード，SDメモリーカードなどの可搬式の記録メディアでは，メディア自体に保護のしくみを備える必要がある．

MP3の拡張フォーマットについて

MP3はもともと音楽データを記録する目的で作られたわけではないので，曲名や演奏者名などのテキスト情報を記録できる部分がありません．MP3ファイルとテキスト・ファイルをセットで保存する方法もありますが，やはり不便です．

〈図 7-11〉 **ID3 Tag ver.1**



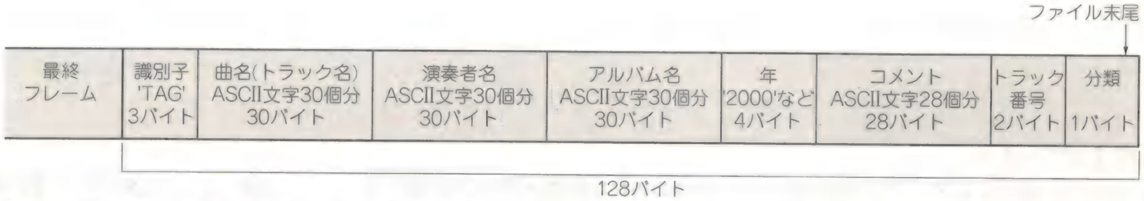
最終フレームの後にデータを付加しても，MP3デコーダはそれを無視するので，ファイルの再生には影響しない．
これを利用して，MP3ファイルの末尾に128バイトのID3 Tagを付加する．

(a) ID3 Tagの付加

0x00	Blues	0x10	Reggae	0x20	Classical	0x30	Ethnic	0x40	Native American	0x50	Folk	0x60	Big Band	0x70	Club
0x01	Classic Rock	0x11	Rock	0x21	Instrumental	0x31	Gothic	0x41	Cabaret	0x51	Folk/Rock	0x61	Chorus	0x71	Tango
0x02	Country	0x12	Techno	0x22	Acid	0x32	Darkwave	0x42	New Wave	0x52	National Folk	0x62	Easy Listening	0x72	Samba
0x03	Dance	0x13	Industrial	0x23	House	0x33	Techno-Industrial	0x43	Psychedelic	0x53	Swing	0x63	Acoustic	0x73	Folklore
0x04	Disco	0x14	Alternative	0x24	Game	0x34	Electronic	0x44	Rave	0x54	Fast-Fusion	0x64	Humour	0x74	Ballad
0x05	Funk	0x15	Ska	0x25	Sound Clip	0x35	Pop-Folk	0x45	Showtunes	0x55	Bebob	0x65	Speech	0x75	Power Ballad
0x06	Grunge	0x16	Death Metal	0x26	Gospel	0x36	Eurodance	0x46	Trailer	0x56	Latin	0x66	Chanson	0x76	Rhythmic Soul
0x07	Hip-Hop	0x17	Pranks	0x27	Noise	0x37	Dream	0x47	Lo-Fi	0x57	Revival	0x67	Opera	0x77	Freestyle
0x08	Jazz	0x18	Soundtrack	0x28	AlternRock	0x38	Southern Rock	0x48	Tribal	0x58	Celtic	0x68	Chamber Music	0x78	Duet
0x09	Metal	0x19	Euro-techno	0x29	Bass	0x39	Comedy	0x49	Acid Punk	0x59	Bluegrass	0x69	Sonata	0x79	Punk Rock
0x0A	New Age	0x1A	Ambient	0x2A	Soul	0x3A	Cult	0x4A	Acid Jazz	0x5A	Avantgarde	0x6A	Symphony	0x7A	Drum Solo
0x0B	Oldies	0x1B	Trip-Hop	0x2B	Punk	0x3B	Gangsta Rap	0x4B	Polka	0x5B	Gothic Rock	0x6B	Booty Bass	0x7B	Acapella
0x0C	Other	0x1C	Vocal	0x2C	Space	0x3C	Top 40	0x4C	Retro	0x5C	Progressive Rock	0x6C	Primus	0x7C	Euro-House
0x0D	Pop	0x1D	Jaz+Funk	0x2D	Meditative	0x3D	Christian Rap	0x4D	Musical	0x5D	Psychedelic Rock	0x6D	Porn Groove	0x7D	Dance Hall
0x0E	R&B	0x1E	Fusion	0x2E	Instrumental Pop	0x3E	Pop/Funk	0x4E	Rock & Roll	0x5E	Symphonic Rock	0x6E	Satire	0x7E	Goa
0x0F	Rap	0x1F	Trance	0x2F	Instrumental Rock	0x3F	Jungle	0x4F	Hard Rock	0x5F	Slow Rock	0x6F	Slow Jam	0x7F	Drum & Bass

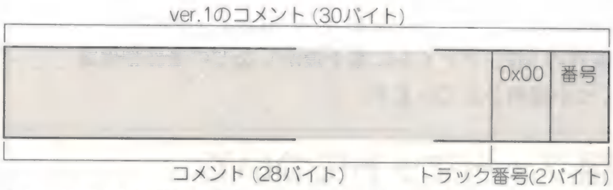
(c) 音楽ジャンルの割り当て（アプリケーションによって若干異なる場合がある）

〈図 7-12〉 **ID3 Tag ver.1.1**



ID3 Tag ver.1のコメントを30バイトから28バイトに減らし，空いた部分にトラック番号2バイトを記録する．

(a) ID3 Tag ver.11の構成



トラック番号記録部は2バイトあるが，1バイト目はかならず0x00，2バイト目にトラック番号を書き込む．
コメントはASCII文字とスペース(0x20)だけなので，29バイト目に0x00を検出したらver.1.1と判断できる．

(b) トラック番号記録部

ファイル末尾

最終フレーム	識別子 'TAG' 3バイト	曲名(トラック名) ASCII文字30個分 30バイト	演奏者名 ASCII文字30個分 30バイト	アルバム名 ASCII文字30個分 30バイト	年 '2000'など 4バイト	コメント ASCII文字30個分 30バイト	分類 1バイト

128バイト

識別子はASCII文字3個，'TAG'.
MP3ファイルの後ろから128バイトを取り出し，その先頭3バイトがASCII文字'TAG'であれば，その128バイトはID3 Tagだと判定できる
曲名，演奏者名，アルバム名，コメントはいずれもASCII文字の30文字分．本来のID3 Tagでは日本語は使用できない．
ただし，現実にはシフトJI..Sなどの日本語コードを埋め込んだMP3ファイルも数多く存在するようだ．その場合は15字まで．
曲名，演奏者名，アルバム名，コメントの空き部分は，すべてスペース(0x20)で埋める．NULL(0x00)や改行(0x0A，0x0D)は使わない．
年は西暦(4桁)をASCII文字4個で表す．
分類は1バイトの符号なし整数値(0～255)に音楽のジャンルが1個ずつ割り当てられている．

(b)　ID3 Tag ver.1の構成

0x80	Club-House	0x90	Thrash Metal	0xA0		0xB0		0xC0		0xD0		0xE0		0xF0	
0x81	Hardcore	0x91	Anime	0xA1		0xB1		0xC1		0xD1		0xE1		0xF1	
0x82	Terror	0x92	JPop	0xA2		0xB2		0xC2		0xD2		0xE2		0xF2	
0x83	Indie	0x93	Synthpop	0xA3		0xB3		0xC3		0xD3		0xE3		0xF3	
0x84	BritPop	0x94		0xA4		0xB4		0xC4		0xD4		0xE4		0xF4	
0x85	Negerpunk	0x95		0xA5		0xB5		0xC5		0xD5		0xE5		0xF5	
0x86	Polsk Punk	0x96		0xA6		0xB6		0xC6		0xD6		0xE6		0xF6	
0x87	Beat	0x97		0xA7		0xB7		0xC7		0xD7		0xE7		0xF7	
0x88	Christian Gangsta Rap	0x98		0xA8		0xB8		0xC8		0xD8		0xE8		0xF8	
0x89	Heavy Metal	0x99		0xA9		0xB9		0xC9		0xD9		0xE9		0xF9	
0x8A	Black Metal	0x9A		0xAA		0xBA		0xCA		0xDA		0xEA		0xFA	
0x8B	Crossover	0x9B		0xAB		0xBB		0xCB		0xDB		0xEB		0xFB	
0x8C	Contenporary Christian	0x9C		0xAC		0xBC		0xCC		0xDC		0xEC		0xFC	
0x8D	Christian Rock	0x9D		0xAD		0xBD		0xCD		0xDD		0xED		0xFD	
0x8E	Merengue	0x9E		0xAE		0xBE		0xCE		0xDE		0xEE		0xFE	
0x8F	Salsa	0x9F		0xAF		0xBF		0xCF		0xDF		0xEF		0xFF	

〈図7-13〉 ID3 Tag ver. 2

ID3 Tag ver.2(略称ID3v2)の現行バージョンであるver.2.3について簡単に紹介する．
ID3v2の詳細については，webサイト(http://www.id3.org/)で仕様書を閲覧できる．

ID3v2ヘッダ 10バイト	ID3v2フレーム 可変長	・・・	ID3v2フレーム 可変長	MP3ビット・ストリーム 可変長

ID3 Tag ver.2

ヘッダの先頭は'ID3'という3個のASCII文字．　MP3ファイルの先頭にこれを検出したら，ID3 Tag ver.2と判定できる．

(a)　基本構成

識別子 'ID3' 3バイト	バージョン 0x03,0x00 2バイト	フラグ xyz00000 1バイト	Tagサイズ 28ビット符号なし整数 4バイト

4個の7ビット値(MSB＝0)を合成して，28ビット符号なし整数として扱う．
これによって，最大のTagサイズは22^{28}＝256Mバイトとなる．

b7＝x，b6＝y，b5＝zが独立した3個のフラグとして扱われる．b4～b0はすべて0．

第1バイトはver.2の中のメジャー・バージョン，第2バイトはリビジョン番号を示す．
この例は，ID3 Tag ver.2.3.0 (バージョン3，リビジョン0)

(b)　ヘッダ

〈図 7-14〉 RIFF WAVE フォーマット

wFormatTag（音声カテゴリ）

タグ(チャンクID) 'RIFF' 4バイト	チャンク長 37+M 4バイト	フォーム・タイプ・コード 'WAVE' 4バイト	タグ(チャンクID) 'fmt' 4バイト	チャンク長 17 4バイト	1 2バイト	nChannels チャネル数 2バイト	nSamplePerSec サンプリング・レート 4バイト

fmtサブチャンク

(a) 本来のRIFF WAVE

wFormatTag（音声カテゴリ）

タグ(チャンクID) 'RIFF' 4バイト	チャンク長 37+M 4バイト	フォーム・タイプ・コード 'WAVE' 4バイト	タグ(チャンクID) 'fmt' 4バイト	チャンク長 17 4バイト	1 2バイト	nChannels チャネル数 2バイト	nSamplePerSec サンプリング・レート 4バイト

fmtサブチャンク

(b) MP3ファイルで使われているRIFF WAVE

そこで，MP3 ファイルにテキスト情報を付加する方法として考案され，デファクト・スタンダードとして普及しているのが **ID3 Tag** です．

一方，p.93 で述べたように，Windows の音声フォーマットとして広く普及しているのが WAVE フォーマット（RIFF WAVE）です．この WAVE フォーマットは RIFF と呼ばれるマルチメディア向けの汎用フォーマットを利用しており，チャンクを付加すればこれらのテキスト情報を記述できます．本来の WAVE フォーマットは PCM 音声データしか扱うことができませんが，この RIFF の形を借用して MP3 データを扱えるようにしたのが RIFF WAVE 形式と RIFF MP3 形式です．

MP3 の拡張フォーマットとして広く使われているものに，もう一つ，**Xing 社**が提唱する VBR（Variable Bit Rate）フォーマットがあります．一般の MP3 ファイルではビット・レートは 128 k ビット/s に固定されているのが普通ですが，MP3 のフレームごとに最適のビット・レートを選択すれば，より良い音質でより小さいファイルを作ることができます．

ID3 Tag

1996 年頃に，MP3 ソフトウェアの Studio3 で採用されたのが始まりだと言われている．まだ MP3 がブームを起こす前のことだ．その後，ほかのソフトウェアにも採用され，また仕様の拡張が提案されたりしながら普及していった．

Xing 社

音声処理技術に定評のあったソフトウェア会社．MP3 エンコーダ/デコーダを製品化するだけでなく，ソースを公開したため，処理エンジンはほかの多くのソフトで採用されている．1999 年に RealNetworks 社に買収された．

● ID3 Tag（Ver. 1）

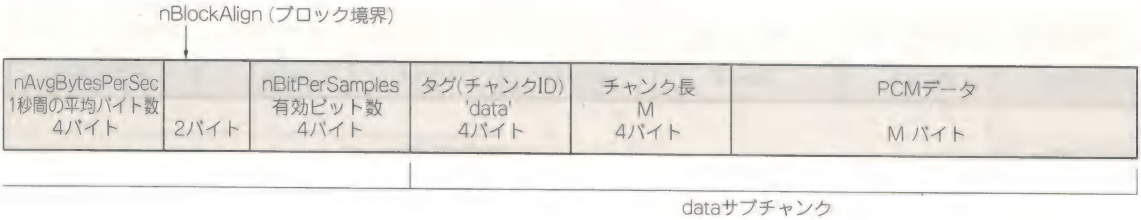
MP3 ファイルでは，最終フレームの後に何か余分なデータを付加したとしても，デコーダは part2_3_length[gr][ch]で計算した最終アドレスの後にあるデータは無視します．**単にファイルサイズが大きくなるだけで**，ファイルの再生に影響はありません．

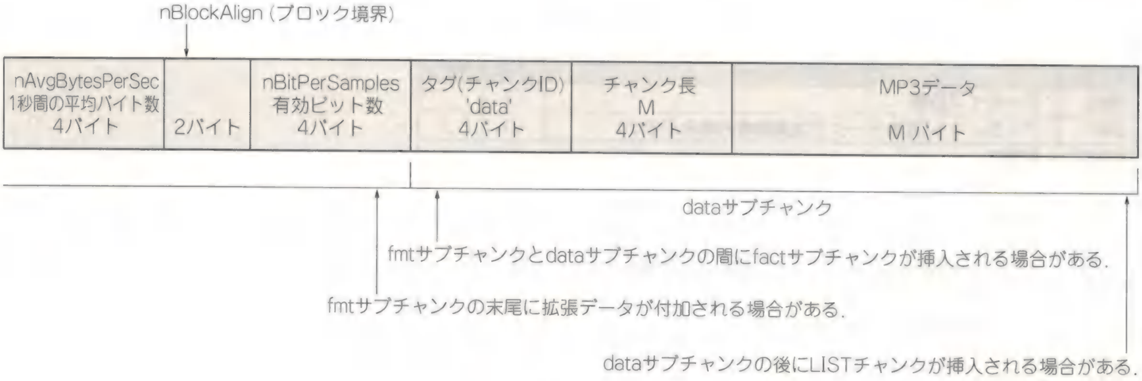
単にファイルサイズが大きくなるだけで

大きくなるといっても，1 分間で約 1M バイトある MP3 ファイルの後ろに 128 バイトぐらい付加してもゴミにもならない．

そこで，MP3 ファイルの末尾に 128 バイトの情報部（タグ）を付加して，曲名（30 バイト），演奏者名（30 バイト），アルバム名（30 バイト），年（4 バイト），任意のコメント（30 バイト），音楽のジャンルの分類（1 バイト）などを記録する方式が提唱されました．この情報部は ID3 Tag と名付けられています(**図 7-11**)．

ID3 Tag は 1996 年頃から少しずつ使われるようになり，1998 年頃から急速に





増え始めたMP3プレーヤ/エンコーダ/デコーダでも採用されるようになりました．また，MP3のデータ・ストリームには手を加えずに，ID3 Tagだけを編集するツールもいろいろ作られるようになりました．

ID3 Tagは3個のASCII文字‘TAG’で始まります．MP3ファイルにID3 Tagが付加されているかどうかは，ファイルの最後の128バイトを取り出し，その先頭3バイトを調べれば判定できます．

● ID3 Tagの拡張

このID3 Tagはきわめて簡単に処理できて，かつ互換性に与える影響も少ないことから，デファクト・スタンダードとして普及しました．しかし，その後ID3 Tagに対して**さらに独自の拡張が**いくつか提唱されており，事情は複雑になってきました．それらと区別するために，もともとのID3 TagはID3 Tag Ver. 1と呼ばれています．

日本製のMP3アプリケーションの一部には，ID3 Tag Ver. 1の独自の拡張として，声優，特撮，アニメなどいくつかの音楽ジャンルを追加したものがあります．とくに新しい名称を付けていないので，他のID3 Tag Ver. 1と区別がつきません．

ID3 Tag Ver. 1.1は，ID3 Tag Ver. 1をわずかに拡張し，コメントを28バイトに減らし，**最後の2バイトに**元のCD-DAのトラック番号を記録できるようにしたものです(**図7-12**)．

ID3 Tag Ver. 2は，MP3ファイルの末尾にタグを付加するのではなく，MP3ファイルの先頭に最大256Mバイトの情報部(タグ)を付加する方式です．文字数の制限がなくなり，またアルバム・ジャケットなどの画像データを記録することもできます．ただし，ID3 Tagという名称は共通ですが，ID3 Tag Ver. 1～1.1とはまったく互換性がありません(**図7-13**)．

さらに独自の拡張が

ID3 Tagのアイデアは良かったが，付加情報が全体で128バイトというのはちょっと遠慮しすぎた．記録できる情報量が限られており，拡張の余地がないのが難点と言える．とくに，日本では曲名や演奏者名が漢字15字分というのは厳しい制限だ．ただし，本来のID3 Tagは漢字コードをサポートしていない．漢字を使うこと自体も独自の拡張の一つだ．

最後の2バイトに

実際には，この2バイトのうち1バイトは0x00（Ver.1.1を示す識別子）を書き込むことになっているので，トラック番号を記録できるのは1バイトだけである．それでも，0～255の数値を記録できるので実用的には困ることはない．

ID3 Tag Ver.2

MP3ビット・ストリームの末尾にこっそりくっついていたVer.1と違って，堂々とした規格になった．情報量が増え，実用上文字数の制限がなくなっただけでなく，Unicodeを採用して各国語をサポートしたこと，歌詞や画像データを記録できることなど特徴が多い．

〈図 7-15〉 LIST チャンク

PCMデータ (MP3データ) Mバイト	タグ(チャンクID) 'LIST' 4バイト	チャンク長 4+(8+P)+(8+Q)+…(8+U) 4バイト	フォーム·タイプ·コード 'INFO' 4バイト	タグ(チャンクID) 'INAM' 4バイト	チャンク長 P 4バイト	チャンク・データ 曲名 Pバイト

dataサブチャンク

INAMサブチャンク

LISTチャンクまたはLISTサブチャンク

LISTチャンクは，RIFFサブチャンクと同様に複数のサブチャンクを内包できる．RIFF DIB，RIFF MIDI，RIFF WAVEなどさまざまなフォーマットで用いることができる．

(a) 構成の例

タグ名	チャンク・データ	本来	MP3
IARL	アーカイブ位置	○	
IART	アーティスト(音楽データでは演奏者の場合が多い)	○	○
ICMS	依頼者	○	△
ICMT	コメント	○	○
ICOP	著作権	○	○
ICRD	作成日	○	○
ICRP	クロップ(周囲部分のトリミング，画像などの場合に使う)	○	
IDIM	寸法(画像データの場合に使う)	○	
IDPI	ドット密度(画像データの場合に使う)	○	
IENG	エンジニア	○	○
IGNR	ジャンル	○	○
IKEY	キーワード	○	△
ILGT	明度(画像データの場合に使う)	○	
IMED	メディア	○	△
INAM	タイトル	○	○
IPLT	パレット(画像データの場合に使う)	○	
IPRD	製品名	○	○
ISBJ	主題	○	△
ISFT	ソフトウェア(ファイル生成に使ったツール名)	○	○
ISHP	シャープネス(画像データの場合に使う)	○	
ISRC	元ファイルの供給元	○	○
ISRF	元データの形式	○	
ITCH	データをディジタル化した技術者	○	△
ILYC	歌詞		△
IMP3	MP3情報		△
IID3	ID3タグ		△

MP3のRIFF WAVE，RIFF MP3でよく使われるタグ．

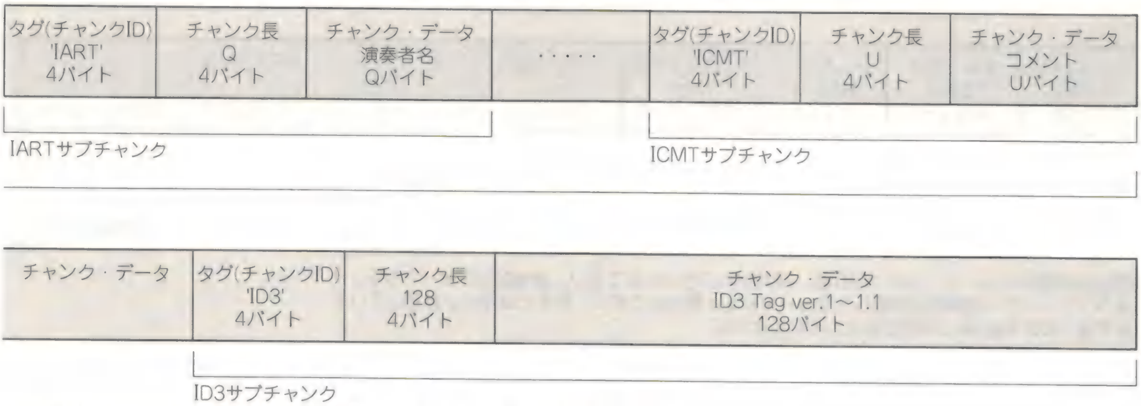
本来のRIFF WAVEで定義されているタグ．

本来のRIFFでは，INFOリストで扱えるタグ(サブチャンク)の種類を規定しており，拡張しないことを推奨している．
だが，MP3ファイルのRIFF WAVE，RIFF MP3ではいくつかの拡張が広く使われている．
さらに，MP3ファイルのRIFF WAVE，RIFF MP3ではこのLISTチャンク部をSIフィールドと呼ぶことが多い．

(b) INFO リストサブチャンク

● RIFF WAVE

本来のWAVEフォーマットは，dataサブチャンクにPCM音声データを収容するように作られています．これを利用して，**dataサブチャンクにMP3のデータ・ストリーム**を収容するようにしたものです．RIFF WAVEという名称は本来のWAVEフォーマットにも使われるので，名称だけからは区別がつきません．また，拡張子も同じWAVのままなので，ファイルの外見上も本来のWAVEファイルと区別がつきません．



RIFF WAVEやRIFF MP3は従来のID3 Tag ver.1～1.1と共存できる．
その代表的な方法が，LISTチャンクの末尾にID3タグを置く方法である．

(c) ID3サブチャンク

RIFFヘッダの部分は本来のWAVEフォーマットと同じです．fmtサブチャンクのwFormatTagの値によって，データがMP3であることを示します．wFormatTag＝1がPCMデータ，wFormatTag＝85(10進)がMP3データです．また，MP3データの場合は，fmtサブチャンクに拡張サイズおよび拡張データという二つのパラメータを付加することができます．また，音声データの全サンプル数を記録するfactサブチャンクを付加する場合もあります(**図7-14**)．

さらに，RIFFにはもともとINAM（データ名），IART（データ作成者），IPRD（製品名）などのさまざまな情報を並列的に記録できる**INFOリスト**という汎用フォーム・タイプが定義されています．タグ名（チャンクID）はLISTなので，LISTチャンクとも呼ばれます．

RIFF WAVEでは，このLISTチャンクをdataサブチャンクの後に付加してテキスト情報を記録できます．これは，SIフィールドと呼ばれる場合もあります(**図7-15**)．

● RIFF MP3（RMP）

やはりRIFFの形式を利用しますが，WAVEフォーム・タイプを使わずに，RMP3という新しいフォーム・タイプを定義して使います(**図7-16**)．

RMP3フォーム・タイプでは，**fmtサブチャンクは省略され**，単にdataサブチャンクにMP3のデータ・ストリームを収容するだけです．拡張子はRMPを用います．さらに，RIFF WAVEと同様にdataサブチャンクの後に**LISTチャンク（SIフィールド）を付加することもできます**．

RIFF MP3はRIFF WAVEよりもファイル構成が簡素であり，拡張子からも本来のWAVEフォーマットとちゃんと区別できるのが利点です．しかし，RIFF MP3対応のプレーヤ/デコーダでしか再生できないのが難点であり，実際にはRIFF WAVEも広く使われています．

RIFF WAVEはきわめてあいまいなフォーマットですが，そのあいまいさのおかげでWAVE/MP3対応のプレーヤで再生できてしまう場合があります．

dataサブチャンクにMP3のデータ・ストリームを

WAVEファイルを扱うアプリケーションは，fmtサブチャンクを参照してデータ形式(PCM)やサンプリング・レート，ビット数を調べて，それをもとにdataサブチャンクにあるPCMデータ・ストリームを順次再生していく．それと同じ方法で，さらにMP3デコーダ機能を追加すればRIFF WAVE形式のMP3データ・ストリームを再生できる．

INFOリスト

RIFFでは，INFOリストに加えられる項目としてINAM，IART，IRRDなどが登録済みのタグ（チャンク名）として定義されている．したがって，たとえばINAM（データ名）であれば，タグ＝'INAM'，チャンク長＝N，チャンク・データ＝'曲名'という構造のサブチャンクがLISTチャンクに加えられる．

fmtサブチャンクは省略され

MP3のデータ・ストリームには，元データのチャネル数，サンプリング・レートなどの情報がヘッダとして含まれている．fmtサブチャンクがなくても再生に支障はないので，RIFF MP3形式ではfmtサブチャンクを省略している．ただし，RIFF WAVE形式では本来のWAVEフォーマットと互換性をもたせるためにfmtサブチャンクもそのまま使っている．

〈図 7-16〉RIFF MP3 フォーマット

タグ(チャンクID)	チャンク長	フォーム・タイプ・コード	タグ(チャンクID)	チャンク長	MP3データ
'RIFF'	12+M	'RMP3'	'data'	M	
4バイト	4バイト	4バイト	4バイト	4バイト	Mバイト

dataサブチャンク

dataサブチャンクの後にLISTチャンクが挿入される場合がある.

構成は他のRIFFフォーム・タイプ(RDIB, RMIDなど)とよく似ており，合理的なフォーマット.
日本のフリーソフト開発者が提唱したフォーマットと言われており，日本ではかなり普及している.
欧米では，ID3 Tag ver.2の方が普及しているようだ.

MP3 の特許ライセンス

ISO/IEC や IEEE などの国際標準には，特定の企業，団体，個人が特許をもつ技術を利用したものが多数あります．特許にかからない公知の技術だけを使ったのでは，性能の低い時代遅れの規格しか作れないからです．

そのため，国際標準に準拠した製品を製造，販売するには，それらの特許権者に対してライセンス契約が必要になる場合も少なくありません．規格を普及させるために事実上ライセンス・フリーでの利用を認める特許権者もありますが，正当な使用料の支払いを要求する特許権者が多いのは当然のことです．ただ，LZW の場合もそうですが，規格が急速に普及したのを見て唐突に使用料を求めるという印象を与えたことから，ユーザの反発を招いている部分が大きいと言えます．

MP3 の場合，ドイツの研究機関である **Fraunhofer IIS-A** が圧縮/伸長技術に関する特許をもっています．そこからライセンスを供与された Thomson Multimedia 社が，MP3 アプリケーションの開発者や MP3 ファイルを頒布する Web サイトの運営者に対してライセンスを供与しているようです．ソフトウェア・メーカや商用 Web サイト，インターネット放送局がおもな対象となっていますが，MP3 エンコーダについてはフリー・ソフトでも使用料が必要だと言われています．

GIF に対する PNG のように，特許ライセンス問題のない完全にフリーな音声フォーマットを普及させようという動きもあります．オープン・ソースを推進する開発者グループが，Vorbis と呼ばれる新しい音声フォーマットを提唱しています．

LIST チャンク（SI フィールド）を付加することもできます．

もともと，RIFF MP3 形式が作られた最大の理由は，ID3 Tag のもつ漢字 15 文字という制限をなくすためだったと言われています．そのため，実際には RIFF MP3 ファイルのほとんどが SI フィールドを使っています．そして，SI フィールドで定義されている項目は，本来の RIFF の INFO リストの項目ではなく，ID3 Tag の項目と互換性をもつように多少拡張されています．

Fraunhofer IIS-A

Fraunhofer Institut Integrierte Schaltungen の略．Fraunhofer グループの中の研究機関で，音声処理技術に定評がある．同社の特許は MPEG-1 Audio や MPEG-2 Audio（とくに Layer3）や MPEG-2 AAC などに使われており，またそれらのエンコーダ/デコーダを製品化している．

第2-1章　PC/AT 互換機のモデム・インターフェースや同期/非同期通信規格

シリアル・ポート

宮崎　仁

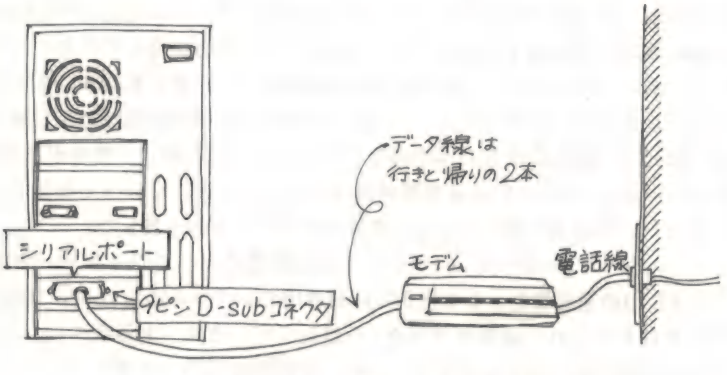
〈名称〉
シリアル・ポート
〈発行日〉
1984年3月
〈発行者〉
International Business Machines Corporation
〈技術資料〉
Technical Reference Personal Computer AT
〈参考資料〉
TIA/EIA-232-F（Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange）
EIA/TIA-562（Electrical Characteristics for An Unbalanced Digital Interface）
EIA/TIA-574（9 Position Non-Synchronous Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange）

IBM社のPC/AT用のシリアル通信アダプタが始まり

シリアル・ポートはもともとIBM PC用の非同期（調歩同期）**シリアル通信アダプタ**として作られました．PC/ATやその後のAT互換機でも，それと同等または上位互換のシリアル・ポートが採用されてきました．

IBM社では，独立した機能モジュールを組み合わせてパソコンを構成するという設計思想から，シリアル・ポートをマザーボードに搭載せず，オプションの拡張ボードとして提供していました．拡張スロットを節約するため，複数の機能をひとつの拡張ボードに搭載することも多く，PC/ATでは2チャネルのシリア

シリアル通信アダプタ
IBM社では拡張ボードのことをアダプタと呼んでいた．



《データの流れがシリアルなのでシリアル・ポートという》

ル・ポートとパラレル・ポート（p. 137参照）を組み合わせたシリアル/パラレル・アダプタが使われていました．その後，シリアル・ポート，パラレル・ポート，FDD/HDDコントローラなどを統合したマルチI/Oアダプタが**サード・パーティ**を中心として数多く作られました．

現在のAT互換機はほとんどすべての機種がシリアル・ポート，パラレル・ポートをマザーボード上に標準装備しています．これは，LSIの高集積化が進んで大部分のI/O機能をチップセットに統合するようになったためです．

サード・パーティ
オリジナルのパソコン・メーカ（この場合はIBM社）に対して，その周辺機器やソフトウェア製品を供給するメーカをサード・パーティと呼ぶ．

シリアル・ポートとRS-232-Cの違い

シリアル・ポートはRS-232-C（現在のバージョンはTIA/EIA-232-F）準拠の非同期シリアル通信ポートです．RS-232-Cは**モデム・インターフェース**の標準規格であり，シリアル・ポートにモデムを接続して，電話回線経由で遠隔地のコンピュータとデータ通信を行うことができます．また，RS-232-Cポートをもつコンピュータや周辺機器を直接接続してデータをやり取りすることもできます．RS-232-C準拠のシリアル通信ポートは，PC/ATに限らずほとんどのパソコンの標準装備となっています．

ただし，現在のパソコンの大部分はRS-232-Cの規格外の高速伝送を行っており，厳密には互換とは言えません．RS-232-Cの伝送速度は最大20kビット/sと定められています．PC/ATの時代のパソコンでは，**シリアル通信ポート**の最大伝送速度は，RS-232-Cの規格内の9.6kビット/sや19.2kビット/sでした．しかし次第に高速化が進み，現在のパソコンでは115.2kビット/sや230.4kビット/sの高速伝送が主流です．

また，コンピュータのシリアル通信ポートどうしを直接接続するような使い方は，やはりRS-232-Cの規格外の使い方です．

PC/ATのシリアル・ポートは，ほかの多くのメーカのRS-232-Cポートとは異なる特徴をもっていました．

ひとつは前述のように，マザーボード上に標準搭載されず，オプションの拡張ボードになっていたことです．この特徴から，PC/ATのユーザはシリアル・ポートの機能を容易にグレードアップできます．IBM社がPC/ATの製造をやめた後でも，サード・パーティから高速，高機能の拡張ボードが次々に供給されてきました．現在のAT互換機はシリアル・ポートを内蔵していますが，ユーザが外付けのシリアル・ポートに置き換えられるように，内蔵シリアル・ポートの回路を切り離せる設計になっているのが普通です．

もうひとつは，ほかの多くのパソコンのRS-232-Cポートは同期/非同期両用で，RS-232-C規格で使われている25ピンD-subコネクタを採用しています．それに対して，PC/ATのシリアル・ポートは非同期専用で，9ピンD-subコネクタを採用している点です．一般のパソコンのデータ通信では同期式はほとんど使いませんが，**BSC**や**SDLC**などの本格的なプロトコルには同期式の機能が必要です．これらの点から，PC/ATの通信機能はほかのパソコンにくらべて貧弱であるとか，RS-232-Cの規格を満たさないと誤解されることもありました．

しかし，シリアル・ポートはPC/ATの通信機能のすべてではなく，IBM社ではPC/AT用の通信機能としてほかに同期専用のシリアル通信ポート（BSC通信アダプタおよびSDLC通信アダプタ）も提供しています．大多数のパソコン・ユーザには同期通信の機能は必要ないため，非同期専用のシリアル・ポートのほうが広く普及しているわけです．

モデム・インターフェース
データ端末装置（この場合はパソコン）をモデムに接続するためのインターフェース．モデムは，ディジタル・データを電話の音声と同等のアナログ信号に変換して，アナログ回線を通じてデータ通信を行うための装置．

シリアル通信ポート
ディジタル・データを1ビットずつ順次送受信する方式をシリアル（直列）通信と呼ぶ．複数のビットを同時に送受信するパラレル（並列）通信にくらべて，時間はかかるが，1本の電線で通信ができるメリットがある．

BSC
Binary Synchronous Communicationsの略．1960年代から1970年代にかけて，これが同期式通信の代表的な方式だった．もともとIBM社が自社のメイン・フレームのために開発した方式．これを元にしてISO国際標準のベーシック手順が作られた．

SDLC
Synchronous Data Link Controlの略．BSCよりも高速かつ確実な通信方式として作られたもので，やはりIBM社が自社のメイン・フレームのために開発した．これを元にしてISO国際標準のHDLCが作られた．

また，もともとのRS-232-Cの規格ではモデム側のコネクタは**25ピンD-sub**が規定されていますが，パソコン側のコネクタの規定はありません（ただし，現行のTIA/EIA-232-Fではパソコン側もモデム側と同じ25ピン・コネクタを用いるように変わっている）．シリアル・ポートの9ピンD-subはRS-232-Cの規格違反ではなく，シリアル・ポートに9ピン-25ピンのモデム・ケーブルを接続すれば，RS-232-C準拠の非同期モデム・インターフェースになります．

なお，IBM社がPC/ATの後継機として発売したPS/2のシリアル・ポートでは，25ピンD-subが採用されました．しかし，大部分のAT互換機は，PC/AT互換の9ピンD-subを採用しています．

25ピンD-sub
一般にRS-232-Cコネクタとして用いられているコネクタ．AT互換機ではパラレル・ポートにも25ピンD-subが使われている．

規格化されたシリアル・ポートEIA/TIA-574

ところが，RS-232-Cの規格を制定した**EIA**では，PC/ATのシリアル・ポートをRS-232-C準拠と呼ぶことは好ましくないと考えました．たしかに，PC/ATが市場で大きなシェアを占めていたアメリカでは，シリアル・ポートをRS-232-Cそのものであると誤解する傾向も広がりはじめていました．

そこで，EIAでは1990年にシリアル・ポートの仕様を新しいEIA規格として採用し，EIA/TIA-574をつくりました．EIA/TIA-574の規格名は「データ端末装置（DTE）とデータ回線終端装置（DCE）の間でシリアル2値データ伝送を行うための9ピン非同期インターフェース」であり，9ピンであることと非同期であることを明記しています．

ただし，EIA/TIA-574は電気的特性をRS-232-Cではなく，新しい規格であるEIA/TIA-562に従うと規定しました．これは，当時のAT互換機のシリアル・ポートの仕様を正しく反映していません．

市場ではすでにシリアル・ポートが事実上の標準であり，あらためて規格を作る必要性がなかったこと，実情に合わない電気的特性を採用したことなどにより，EIA/TIA-574はほとんど普及せずに終わりました．

EIA
Electronic Industries Allianceの略．一般にはアメリカ電子工業会と呼ばれる．昔から標準化に熱心に取り組んでおり，RS（Recommended Standard）と呼ばれる業界標準を数多く制定している．

マイクロソフトのPC規格　PC 95からPC 98へ

Microsoft社が提唱したPC 95ハードウェアデザインガイドでは，当時のAT互換機の状況を反映して，**16550 A**互換で115.2kビット/s対応のシリアル・ポートが必須（または推奨）条件となっています．

PC 97とPC 98では，新しいシリアル・インターフェースとしてUSBが必須になり，シリアル・ポートは任意装備のデバイスとなりました．しかし，市場にはまだシリアル・ポート用の周辺機器のほうが多く出回っており，PC用チップセットにもシリアル・ポートの機能が内蔵されていることから，AT互換機でシリアル・ポートを装備しないものは作られていません．

しかし，今後USB用の周辺機器が多く出回るようになれば，数年後にはシリアル・ポートは次第に姿を消していくことも考えられます．

16550A
PC/ATで使われていたシリアル通信コントローラはNS社が開発した8250/16450だった．16550 Aはその改良版．

非同期モデム，パソコンどうし，シリアル通信のための取り決め

PC/ATのシリアル・ポートは**RS-232-C準拠**の非同期モデム・インターフェース機能をもっています．実際には，モデム以外の周辺機器との接続やパソコンどうしの接続にも用いられますが，それはRS-232-Cの規格外の使い方です．ま

RS-232-C準拠
準拠とは標準に完全に従っているということ．正会員，準会員などというときの準とは意味が違う．

た，オリジナルのシリアル・ポートは最大伝送速度が9.6 k ビット/s で，RS-232-C の規格を満たしています．しかし，現在の AT 互換機のシリアル・ポートは，RS-232-C の規格外の高速伝送に使われています．

シリアル・ポートの仕様は，外側から見れば RS-232-C にほぼ従い，内部はシリアル通信コントローラの INS8250/NS16450 の仕様に従っています．

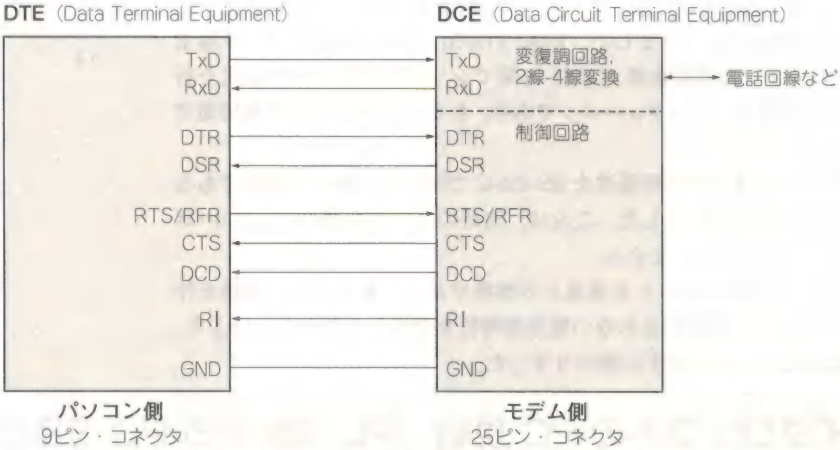
非同期モデム・インターフェース

DCE

Data Circuit terminating Equipment の略．Data Circuit（データ回線）とは電話回線のことで，電話局側から見れば，電話回線の末端にモデムがぶら下がっているように見えることから付けられた呼び名．ユーザ側から見れば，終端（terminating）というより，データ回線への入り口と考える方が自然だろう．

RS-232-C では，パソコンのように主体的にデータ処理を行う機器を DTE（データ端末装置），モデムのようにデータを電話回線に中継する機器を **DCE**（データ回線終端装置）と呼びます（**図 1-1**）．DTE と DCE の間でシリアル・データを双方向にやり取りするために，送信データ（TxD）と受信データ（RxD）の 2 本のデータ線を使います．また，データ転送を制御するためのハンドシェイク線として，送信要求/受信可（RTS/RFR），送信可（CTS），データ端末レディ（DTR），

〈図 1-1〉非同期モデム・インターフェース



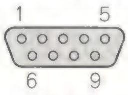
同じ信号名どうしを接続する（ストレート・ケーブル），使用しないピン（25ピン・コネクタ側）はオープン

(a) 接続の概要

EIA-574

ピン番号	I/O	信号名	備　考
1	I	DCD	キャリア検出
2	I	RxD	受信データ
3	O	TxD	送信データ
4	O	DTR	データ端末レディ
5	—	GND	グラウンド
6	I	DSR	データ・セット・レディ
7	O	RTS	送信要求
8	I	CTS	送信可
9	I	RI	被呼表示

コネクタ形状/D-Sub9ピン（オス）

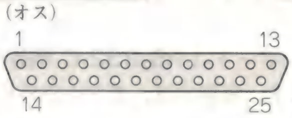


EIA-232

ピン番号	I/O	信号名	備　考
1	—	(N.C.)	
2	O	TxD	送信データ
3	I	RxD	受信データ
4	O	RTS	送信要求
5	I	CTS	送信可
6	I	DSR	データ・セット・レディ
7	—	GND	グラウンド
8	I	DCD	キャリア検出
9	—	(N.C.)	
10	—	(N.C.)	
11	—	(N.C.)	
12	—	(N.C.)	
13	—	(N.C.)	

ピン番号	I/O	信号名	備　考
14	—	(N.C.)	
15	—	(N.C.)	
16	—	(N.C.)	
17	—	(N.C.)	
18	—	(N.C.)	
19	—	(N.C.)	
20	O	DTR	データ端末レディ
21	—	(N.C.)	
22	I	RI	被呼表示
23	—	(N.C.)	
24	—	(N.C.)	
25	—	(N.C.)	

コネクタ形状/D-Sub25ピン（オス）



(b) 9ピン・コネクタと25ピン・コネクタ

データ・セット・レディ(DSR)，データ・キャリア検出(DCD)の5本を使います．

この7本のデータ/制御線にグラウンド線を加えた8本が，非同期モデム・インターフェースのために最低限必要です．

シリアル・ポートでは，さらにDCEが電話回線から呼び出し信号を受けたことをDTEに通知するための信号線として，被呼表示(RI)をサポートしています．これらの9本の線を9ピンD-subコネクタに割り当てています．

DCE側のコネクタは，RS-232-Cの規格で25ピンD-subと定められています．そのため，シリアル・ポートとモデムの接続には9ピン-25ピンの接続ケーブルを使います．RS-232-Cの信号名は，DTE側で出力の信号はDCE側では入力，DTE側で入力の信号はDCE側では出力となるように決められており，DTEとDCEの同じ信号名どうしを接続するようになっています．すなわち，接続ケーブルはストレート・ケーブルになります．

パソコンどうしの接続

パソコンどうしの接続はRS-232-Cの規格外ですが，実際には広く行われています(**図1-2**)．この場合，両方がPC/ATやAT互換機であれば，**9ピン-9ピンの接続ケーブル**で接続することになります．また，DTEどうしの接続の場合，同じ信号名どうしを接続すると出力と出力，入力と入力がぶつかってしまいますから，TxDとRxD，RTSとCTS，DTRとDSRを入れ換えて接続するクロス・ケーブルを用います．

また，相手側のパソコンが25ピン・コネクタのRS-232-Cポートをもつ場合は，9ピン-25ピンのクロス・ケーブルで接続することになります．

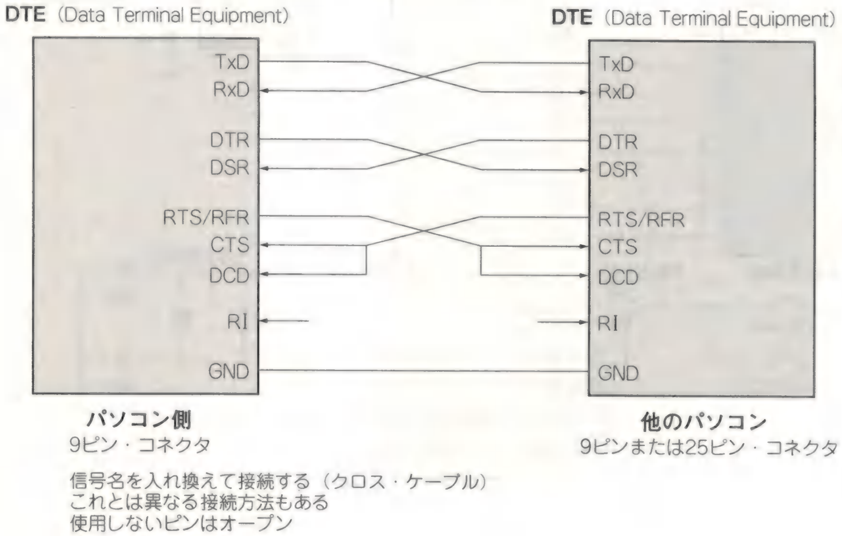
9ピン-9ピンの接続ケーブル
シリアル・ポートの場合，パソコンとモデムの接続にはかならず9ピン-25ピンのストレート・ケーブルを使う．9ピン-9ピンのケーブルは常にパソコンどうしの接続用であり，かならずクロス・ケーブルになる．

シリアル通信コントローラ

PC/AT用のシリアル/パラレル・アダプタでは，NS社のINS8250/NS16450が使われていました．16450は8250の選別品で，特性は少し優れていますが，機

ボーレート・ジェネレータ
高速のクロックを分周して，通信に必要とされるスピードのクロックを生成する回路．

〈図1-2〉パソコンどうしの接続



FIFO

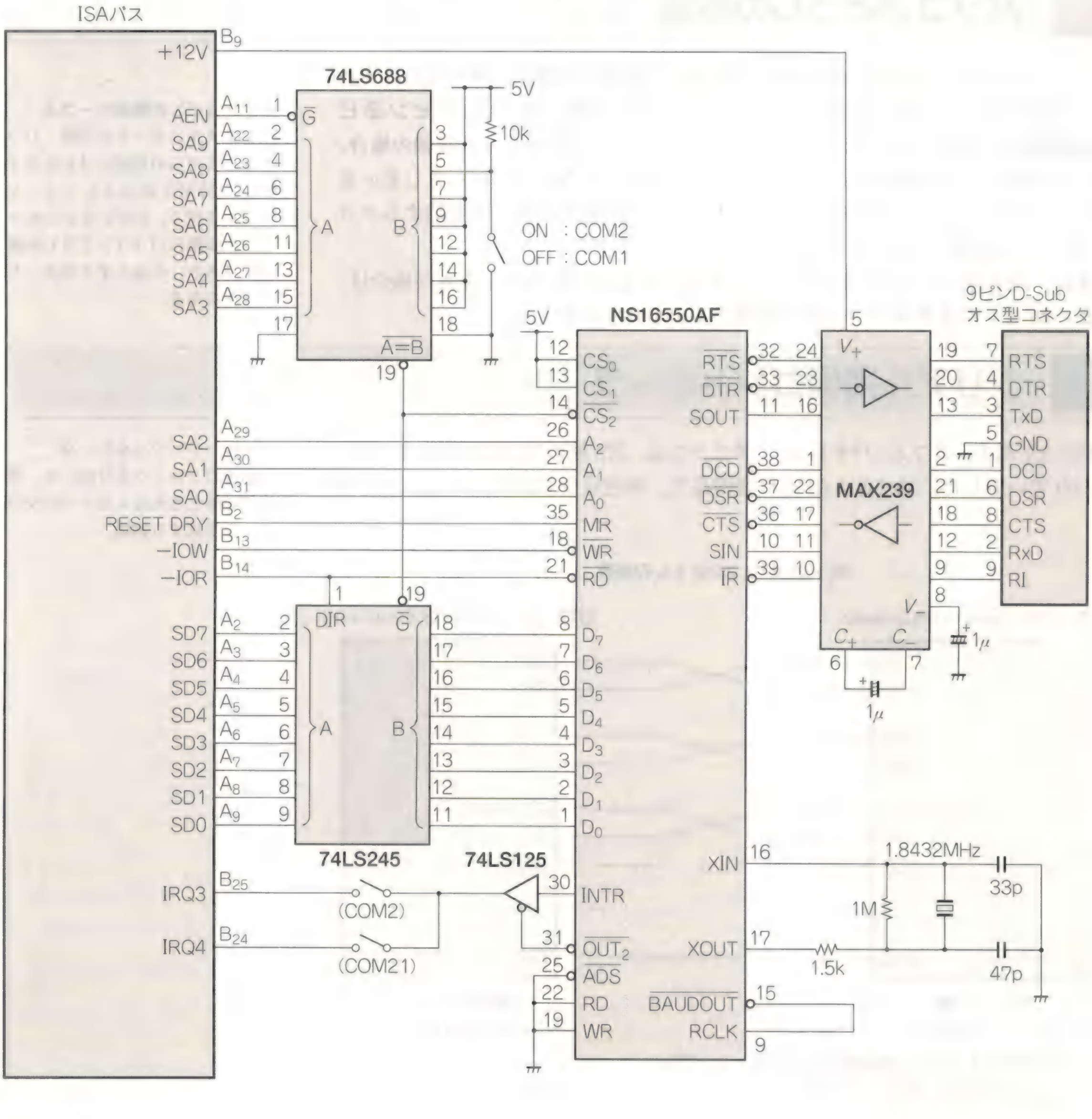
First-In First-Out の略．先入れ先出しメモリとも呼ばれる．データ通信では，通信コントローラ部分に FIFO を装備することによって，通信の相手側から送られてきたデータを一時的に蓄えたり，CPU から書き込まれたデータを一時的に蓄えるバッファ・メモリとして利用できる．このバッファ・メモリは，通信の相手側と，CPU 側との伝送速度の違いを吸収する働きがある．

能的には同等です．

8250/16450 は非同期専用のシリアル通信コントローラで，送受信クロック生成用の**ボーレート・ジェネレータ**を内蔵しています．最大伝送速度は 56 k ビット/s です．PC/AT では 8250/16450 を 2 個使って，2 チャネルのシリアル・ポートを実現していました．

その後，8250/16450 の上位互換でより高速動作が可能な NS16550A が使われるようになりました（**図 1-3**）．16550 A は送信バッファ，受信バッファとしてそれぞれ 16 バイトの **FIFO** を内蔵しており，送受信における CPU の負担を大幅に減らすことができます．また，動作速度そのものも高速化されており，256 k ビット/s の最大伝送速度が得られます．Microsoft 社が Windows パソコンのハードウェアのガイドラインとして提唱している PC 95 や PC 98 でも，シリア

〈図 1-3〉シリアル通信コントローラ



ル・ポートは16550 A互換であることが必要とされています．

現在では，16550 Aのさらに上位互換のシリアル通信コントローラや，パラレル・ポートなどと組み合わせたマルチI/Oチップが使われています．大部分のAT互換機では，16550 A相当の機能を組み込んだチップセットが使われています．

PC 98システムデザインガイドでも，シリアル・ポートの装備は任意ですが，装備するなら16550 A互換が必須条件となっています．

8250/16450/16550Aのレジスタ構成

8250/16450やその上位互換の16550 Aは，3本のアドレス線で選択される**8個のレジスタ**をもっています．また，別にボーレート・ジェネレータの**分周値設定用の2個のラッチ**があります（**表1-1**）．

8個のレジスタ
後述のように，PC/ATではI/Oアドレス03 F8h ～ 03 FFh（チャネル1），02 F8h ～ 02 FFh（チャネル2）に割り付けられている．

分周値設定用の2個のラッチ
レジスタを通じてデータを書き込むので，I/Oアドレスは割り当てられていない．

レジスタ0は送信バッファ（TBR）/受信バッファ（RBR）です．また，レジスタ3の分周器ラッチ・ビット（DLAB）をセットしたときは，分周値ラッチ下位（DLL）となります．

レジスタ1は割り込み許可レジスタ（IER）で，割り込み要因に応じて4レベルの割り込みを個別にイネーブル/ディセーブルできます．割り込み要因は受信ステータス割り込みがもっとも優先度が高く，次いで受信割り込み，送信割り込み，モデム・ステータス割り込みの順となります．レジスタ3の分周器ラッチ・ビット（DLAB）をセットしたときは，分周値ラッチ上位（DLM）となります．

レジスタ2は割り込みIDレジスタ（IIR）で，割り込み発生時に割り込み要因を保持しています．割り込みIDレジスタは読み出し専用ですが，16550 Aでは同じアドレスに書き込み用のFIFO制御レジスタ（FCR）を追加しています．

レジスタ3はライン制御レジスタ（LCR）です．データ長（5/6/7/8ビット），ストップ・ビット長（1/1.5/2ビット），パリティ・ビット（あり/なし，奇数/偶数/1に固定），ブレーク送信など，非同期通信に必要な各種のパラメータを設定できます．

レジスタ4はモデム制御レジスタ（MCR）で，モデム制御線のRTSとDTR，汎用ビット出力のOUT1とOUT2のハイ/ローを個別に切り替えられます．PC/ATのシリアル・ポートでは，OUT1は使用されませんが，OUT2は割り込み信号のマスクに使われます．

レジスタ5はライン・ステータス・レジスタ（LSR）です．送信/受信の状態やエラー発生などのステータスを読み出すことができます．

レジスタ6はモデム・ステータス・レジスタ（MSR）です．モデム制御線のCTS，DSR，DCD，RIの現在の状態が読み出せるだけでなく，前回の読み出し後に状態の変化があったかどうかを読み出せます．

レジスタ7はスクラッチ・レジスタ（SCR）です．これは単なる8ビットのメモリで，任意の目的に使用できます．

オリジナルのPC/ATでは，シリアル・ポート1のI/Oアドレスを03 F8h ～ 03 FFh，シリアル・ポート2のI/Oアドレスを02 F8h ～ 02 FFhに割り付けてあります．また，シリアル・ポート1の割り込みチャネルはIRQ4，シリアル・ポート2の割り込みチャネルはIRQ3に割り付けられています．

最近のAT互換機では，シリアル・ポートを4チャネルに拡張できるものが多くなっていますが，割り込みチャネルはIRQ4とIRQ3を共用します．すなわち，シリアル・ポート1とシリアル・ポート3の割り込みチャネルはIRQ4，シリアル・ポート2とシリアル・ポート4の割り込みチャネルはIRQ3になります．

〈表 1-1〉 8250/16450/16550A のレジスタ構成

制御ビット DLAB	アドレス線			8250/16450 のレジスタ		16550A のレジスタ	
	A_2	A_1	A_0	リード	ライト	リード	ライト
0	0	0	0	受信バッファ (RBR)	送信バッファ (TBR)	受信バッファ (RBR)	送信バッファ (TBR)
0	0	0	1	割り込み許可 (IER)		割り込み許可 (IER)	
×	0	1	0	割り込み ID (IIR)		割り込み ID (IIR)	FIFO 制御 (FCR)
×	0	1	1	ライン制御 (LCR)		ライン制御 (LCR)	
×	1	0	0	モデム制御 (MCR)		モデム制御 (MCR)	
×	1	0	1	ライン・ステータス (LSR)		ライン・ステータス (LSR)	
×	1	1	0	モデム・ステータス (MSR)		モデム・ステータス (MSR)	
×	1	1	1	スクラッチ (SCR)		スクラッチ (SCR)	
1	0	0	0	分周値ラッチ下位 (DLL)		分周値ラッチ下位 (DLL)	
1	0	0	0	分周値ラッチ上位 (DLM)		分周値ラッチ上位 (DLM)	

(a) レジスター覧

レジスタ	方向	b7 (MSB)	b6	b5	b4
受信バッファ (RBR)	R	b7	b6	b5	b4
送信バッファ (TBR)	W	b7	b6	b5	b4
割り込み許可 (IER)	R/W	0	0	0	0
割り込み ID (IIR)	R	〈FIFO 動作ステータス〉 00 ： FIFO は非動作 11 ： FIFO は動作可		0	0
〈FIFO 制御 (FCR)〉	〈W〉	〈受信バッファ・トリガ・レベル〉 11 ： 14 バイト 10 ： 8 バイト 01 ： 4 バイト 00 ： 1 バイト		〈予約〉	〈予約〉
ライン制御 (LCR)	R/W	分周値ラッチ (DLAB) 1：分周値ラッチにアクセス 0：レジスタにアクセス	ブレーク制御 1：ブレーク送出 0：通常出力	固定パリティ・ビット 1：固定値 0：通常パリティ	偶数パリティ 1：偶数パリティ 0：奇数パリティ
モデム制御 (MCR)	R/W	0	0	0	ループ・テスト 1：ローカル・ループ・テスト 0：通常動作
ライン・ステータス (LSR)	R/W	〈受信 FIFO エラー〉 1：受信 FIFO でエラー発生 (LSR 読み出しで 0 に戻る) 0：エラーなし	送信アンダ・ラン 1：送信シフトレジスタ空 (データ書き込みで 0 に戻る) 0：送信シフトレジスタ空でない	送信バッファ空 1：送信バッファ空 (データ書き込みで 0 に戻る) 0：送信バッファ空でない	ブレーク受信 1：ブレーク受信あり (LSR 読み出しで 0 に戻る) 0：ブレーク受信なし
モデム・ステータス (MSR)	R/W	DCD 1：/DCD 入力アクティブ (Lo) 0：/DCD 入力非アクティブ (Hi)	RI 1：/RI 入力アクティブ (Lo) 0：/RI 入力非アクティブ (Hi)	DSR 1：/DSR 入力アクティブ (Lo) 0：/DSR 入力非アクティブ (Hi)	CTS 1：/CTS 入力アクティブ (Lo) 0：/CTS入力非アクティブ (Hi)
スクラッチ (SCR)	R/W	b7	b6	b5	b4
分周値ラッチ下位 (DLL)	R/W	b7	b6	b5	b4
分周値ラッチ上位 (DLM)	R/W	b15	b14	b13	b12

割り込み ID ビットと
割り込み保留ビット

割り込み ID (8250/16450 は b2 ～ b1、16550A は b3 ～ b1)			割り込み保留
b3	b2	b1	b0
0	0	0	1
0	1	1	0
0	1	0	0
1	1	0	0
0	0	1	0
0	0	0	0

(b) 各レジスタのビット構成（〈〉内は 16550A のみ）

b3	b2	b1	b0（LSB）
b3	b2	b1	b0
b3	b2	b1	b0
モデム・ステータス割り込み 1：割り込み許可 0：割り込みマスク	受信ステータス割り込み 1：割り込み許可 0：割り込みマスク	送信割り込み 1：割り込み許可 0：割り込みマスク	受信割り込み 1：割り込み許可 0：割り込みマスク
〈割り込みID〉 下表参照	割り込みID 下表参照		割り込み保留 下表参照
〈DMAモード〉 1：DMAはバースト転送可 0：DMAはシングル転送のみ	〈送信FIFOリセット〉 1：リセットする （リセット後は0に戻る） 0：リセットしない	〈受信FIFOリセット〉 1：リセットする （リセット後は0に戻る） 0：リセットしない	〈FIFOイネーブル〉 1：FIFO動作可にする 0：FIFO非動作にする
パリティ・イネーブル 1：送受信パリティあり 0：送受信パリティなし	ストップ・ビット長 1：2ビット（6～8ビット・データ） 1：1.5ビット（5ビット・データ） 0：1ビット	データ長 11：8ビット 10：7ビット 01：6ビット 00：5ビット	
OUT2 1：/OUT2出力アクティブ(Lo) 0：/OUT2出力非アクティブ(Hi)	OUT1 1：/OUT1出力アクティブ(Lo) 0：/OUT1出力非アクティブ(Hi)	RTS 1：/RTS出力アクティブ(Lo) 0：/RTS出力非アクティブ(Hi)	DTR 1：/DTR出力アクティブ(Lo) 0：/DTR出力非アクティブ(Hi)
フレーミング・エラー 1：フレーミング・エラー発生 (LSR読み出しで0に戻る) 0：エラーなし	パリティ・エラー 1：パリティ・エラー発生 (LSR読み出しで0に戻る) 0：エラーなし	受信オーバラン 1：オーバラン・エラー発生 (LSR読み出しで0に戻る) 0：エラーなし	受信データ有効 1：受信バッファに有効データあり (データ読み出しで0に戻る) 0：受信バッファに有効データなし
DCD変化 1：/DCD入力変化あり 0：/DCD入力変化なし	RI後エッジあり 1：/RI入力Lo→Hi変化あり 0：/RI入力Lo→Hi変化なし	DSR変化 1：/DSR入力変化あり 0：/DSR入力変化なし	CTS変化 1：/CTS入力変化あり 0：/CTS入力変化なし
b3	b2	b1	b0
b3	b2	b1	b0
b11	b10	b9	b8

優先順位	割り込み要因
-	割り込みなし
最高	受信ステータス割り込み
	受信割り込み
	〈FIFOタイムアウト割り込み〉
	送信割り込み
最低	モデム・ステータス割り込み

シリアル・ポートの伝送速度

シリアル・ポートの伝送速度を制限する要因には，RS-232-C の電気的仕様からくるもの，シリアル通信コントローラの仕様からくるもの，パソコンの BIOS や割り込み処理の仕様からくるものなどがあり，複雑にからみあっています．

シリアル通信コントローラとして 8250/16450 を使っていた頃は，次のように伝送速度の上限が決まっていました．

8250/16450 の送受信クロックは，基準クロックを内蔵ボーレート・ジェネレータで分周して作ります．送信，受信とも，伝送速度の 16 倍の送受信クロックを用います（**表 1-2**）．

PC/AT のシリアル・ポートでは，基準クロックとして 1.8432 MHz を使っていました．384 分周で 300 ビット/s, 192 分周で 600 ビット/s, …，12 分周で 9.6 k ビット/s というように，モデム・インターフェースに必要な 300 ビット/s の**倍数系列の伝送速度**が得られます．PC/XT や PC/AT などの初期のパソコンは CPU の処理速度が遅く，また BIOS で割り込みをサポートしていないためあまり高速の伝送ができません．最大伝送速度は 9.6 k ビット/s に制限されていました．

その後の AT 互換機では，パソコンの処理速度の向上によって，最大伝送速度 19.2 k ビット/s や 38.4 k ビット/s, 56 k ビット/s などの製品が作られるようになりました．8250/16450 自体の最大伝送速度は 56 k ビット/s ですが，300 ビット/s の倍数系列なら 38.4 k ビット/s（3 分周）まで使うことができます．RS-232-C の規格からすると 38.4 k ビット/s や 56 k ビット/s は規格外ですが，伝送距離が短ければ**従来のドライバ/レシーバ IC** でも規格外の高速伝送が可能です．次第に規格外の高速製品が増えてきたことから，ドライバ/レシーバ IC も 64 k ビット/s 程度の伝送速度を保証したものが増えてきました．

さらに，16 バイトの送受信 FIFO を備えた 16550 A を用いることによって，より高速な伝送が可能になりました．16550 A は最大 256 k ビット/s に高速化されており，1.8432 MHz の基準クロックで 115.2 k ビット/s（1 分周）まで使うことができます．それ以上の伝送速度（たとえば 230.4 k ビット/s）に設定する場合は，より高速の基準クロックが必要です．16550 A には最大 8 MHz の基準クロックを入力することができます．

この，112.5 k ビットはそれからしばらくの間，AT 互換機のシリアル・ポートの標準的な最大転送速度と見なされてきました．ドライバ/レシーバ IC でも 115.2 k ビット/s をサポートするものが多くなりました．PC 95 ハードウェアデザインガイドでは，シリアル・ポートは 115.2 k ビット/s 対応であることが必須

倍数系列の伝送速度

もっとも初期に普及したモデム規格は，300 ビット/s の V. 21 である．この規格では，RS-232-C の部分のデータ速度も 300 ビット/s に合わせることが必要であり，そのため一般のパソコンの RS-232-C ポートでも 300 ビット/s の倍数系列の伝送速度を使うようになった．その後，モデムの伝送速度は 1200 ビット/s の V. 22, 2400 ビットの V. 22 bis, 9600 ビット/s の V. 32, 14.4 k ビット/s の V. 32 bis, 28.8 k ビットの V. 34, 56 k ビットの V.90 と高速化されていった．これらのモデムは内部に大容量のバッファ・メモリをもつため，RS-232-C 部分のデータ速度はモデムと一致している必要はなくなった．そのため，通常はそのパソコンで使える最高速度に設定するようになった．

従来のドライバ/レシーバ IC

古くからある RS-232-C 用ドライバ/レシーバ IC は，RS-232-C 規格に合わせて最大 20 k ビット/s の伝送レートを保証している．ただし，実際にはそれ以上のスピードで動作するものもある．

トランジスタ技術 SPECIAL No.51　エレクトロニクスの基礎と実用技術を満載したフィールド・ワーク・マガジン
B5 判 160 頁
定価 1,835 円（税込）
特集 データ通信技術基礎講座
RS232C の徹底理解からローカル通信の実用技術まで
こうパソコンの台数が増えてくると，パソコン同士を接続したり，モデムやプロッタといった周辺機器を接続することがあたりまえになってきました．これらの機器同士を物理的にケーブルで接続しただけでは，データ通信はできません．本書は，この通信で非常によく使われているシリアル伝送の規格 RS232C について，やさしく解説します．さらに一般の電話回線を利用してデータを送るための装置 モデムを取り上げ，その成り立ちとハードウェアの構成を紹介します．最後に，パソコン同士を有機的に接続する LAN の構成法についても言及します．
CQ出版社　〒170-8461 東京都豊島区巣鴨 1-14-2 販売部 TEL（03）5395-2141 振替 00100-7-10665

〈表 1-2〉8250/16450/16550A の伝送速度

基準クロック[MHz]	分周値	16倍クロック[kHz]	伝送速度[ビット/s]	8250/16450	16550A
1.8432	384	4.8	300	○	○
	192	9.6	600	○	○
	96	19.2	1200	○	○
	48	38.4	2400	○	○
	24	76.8	4800	○	○
	12	153.6	9600	○	○
	6	307.2	19200	○	○
	3	614.4	38400	○	○
	2	921.6	57.6k	○	○
	1	1843.2	115.2k	×	○
3.072	640	4.8	300	○	○
	320	9.6	600	○	○
	160	19.2	1200	○	○
	80	38.4	2400	○	○
	40	76.8	4800	○	○
	20	153.6	9600	○	○
	10	307.2	19200	○	○
	5	614.4	38400	○	○
	4	768	48k	○	○
	3	1024	64k	×	○
7.3728	1536	4.8	300	×	○
	768	9.6	600	×	○
	384	19.2	1200	×	○
	192	38.4	2400	×	○
	96	76.8	4800	×	○
	48	153.6	9600	×	○
	24	307.2	19200	×	○
	12	614.4	38400	×	○
	8	921.6	57.6k	×	○
	6	1228.8	76.8k	×	○
	4	1843.2	115.2k	×	○
	2	3686.4	230.4k	×	○
	1	7372.8	460.8k	×	×

となっています．

最近では，さらに高速の 230.4 k ビット/s に対応できるドライバ/レシーバ IC がいくつか発売され，パソコンでも 230.4 k ビット/s をサポートする製品が増えています．

シリアル・ポートの電気的仕様

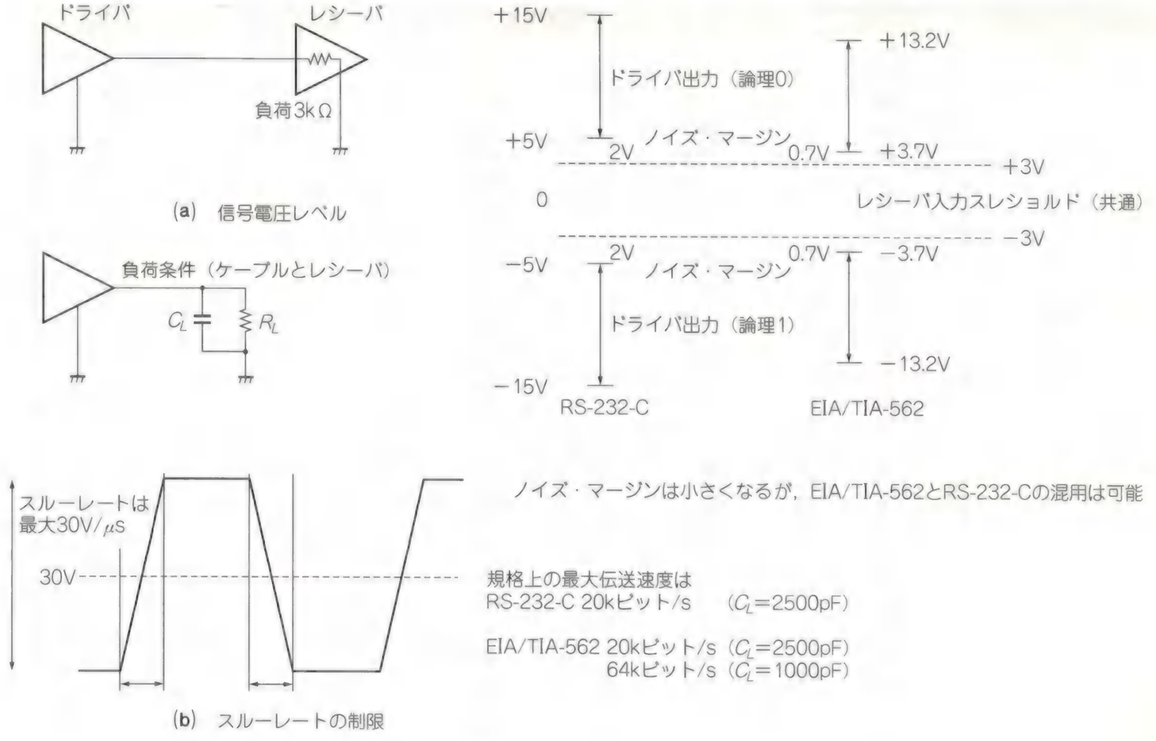
PC/AT のシリアル・ポートは RS-232-C の電気的仕様に準拠しています（**図 1-4**）．オリジナルのシリアル/パラレル・アダプタの回路では，ドライバ/レシーバ IC として**± 12 V 電源が必要**な SN75150/75154 が使われていました．最近では，MAX232 に代表される電圧コンバータ内蔵の単電源動作ドライバ/レシーバ IC が一般的です．シリアル・ポートの仕様に合せたドライバ× 3（TxD，RTS，DTR），レシーバ× 5（RxD，CTS，DSR，DCD，RI）の IC がよく使われています．

いずれにせよ，RS-232-C 用ドライバ/レシーバを使えば，20 k ビット/s 以下の伝送を行う限り，RS-232-C に準拠した電気的仕様を実現できます．ただし，一部のノート・パソコンなどでは RS-232-C ではなく，後述の EIA/TIA-562 に準拠

± 12 V 電源が必要

RS-232-C の規格上の信号電圧は，最小± 5 V，最大± 15 V と規定されている．昔のコンピュータは± 12 V 電源を備えたものが多かったので，RS-232-C 用 IC にも± 12 V 電源を供給するのが普通だった．その後，IC 内部に＋5 V →＋ 10 V の 2 倍昇圧回路と＋ 10 V →− 10 V の反転回路を内蔵した MAX232 が発売され，5 V 単電源動作の RS-232-C 用 IC として大ヒットした．

〈図 1-4〉電気的仕様



したものもあるようです．

規格外の高速伝送を行うためには，ドライバ/レシーバ IC にも高速のものを用いなければなりません．RS-232-C 用ドライバ/レシーバは，初期の製品は最大 20 k ビット/s の規格の範囲だけで動作を保証していましたが，その後 64 k ビット/s, 120 k ビット/s, 250 k ビット/s などの高速製品も作られるようになりました．従来の RS-232-C との互換性を失わないように，負荷条件，信号レベル，スルーレートなどの仕様は RS-232-C の規定に反しないように作られています．とくに問題となるのは，RS-232-C では信号の反射による波形歪みを防ぐために，スルーレートを最大 30 V/μs に制限している点です．この 30 V/μs の制限を守って高速化を進めると，だいたい 250 k ビット/s が上限となります．

EIA/TIA-562 は，低電圧システム向けにドライバの出力振幅を小さくしたインターフェース規格です．また，近距離の場合に限って伝送速度は最大 64 k ビット/s まで認めています．それ以外の部分は RS-232-C と同等または類似の仕様であり，RS-232-C は満たしませんが，実用上は相互に接続が可能です．EIA/TIA-562 用ドライバ/レシーバもいくつか製品化されており，一部のノート・パソコンなどで使われているようです．

スルーレートを最大 30 V/μs に制限

信号の立ち上がり，立ち下がりのスピードが速いと，電線を伝播する信号が反射を繰り返して波形歪みが大きくなる．それを防ぐには，立ち上がり，立ち下がりのスルーレートを制限するか，信号が反射しないように電線の両端でインピーダンスの整合をとる（いわゆる終端）かのどちらかが必要．RS-232-C ではスルーレートを制限して反射を防いでおり，そのかわり高速の伝送は難しくなってしまった．

第2-2章 PC/AT互換機のためのプリンタ・インターフェース規格

パラレル・ポート(IEEE 1284)

宮崎　仁

〈名称〉
パラレル・ポート
〈発行日〉
1984年3月(パラレル・ポート)
1994年12月(IEEE 1284)
〈発行者〉
International Business Machines Corporation (パラレル・ポート)
IEEE (IEEE 1284)
〈技術資料〉
Technical Reference Personal Computer AT
IEEE 1284-1994 Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers

IBM社のPC用セントロニクス互換プリンタ・インターフェース

パラレル・ポートはもともとIBM PC用の**セントロニクス互換プリンタ・インターフェース**として作られました．PC/ATやその後のAT互換機でも，それと同等または上位互換のパラレル・ポートが採用されてきました．パラレル・ポートは回路規模が小さいことから，ほかの機能と組み合わせた複合アダプタがいろいろ作られました．たとえば，モノクローム・アダプタにパラレル・ポートを内蔵したり，2チャネルのシリアル・ポートとパラレル・ポートを組み合わせたシリアル/パラレル・アダプタが作られたりしました．

現在のAT互換機はほとんどすべての機種がシリアル・ポート，パラレル・ポートをマザーボード上に標準装備しています．これは，LSIの高集積化が進んで大部分のI/O機能をチップセットに統合するようになったためです．

一般に，セントロニクス互換のプリンタ・ポートは，パソコンからプリンタに向けてデータを出力するだけの**単方向ポート**となっています．PC/AT用のパラレル・ポートは，一般のセントロニクス互換のプリンタ・ポートと少し異なっていて，現在のデータ線の出力状態をモニタする機能がついていました．ただし，出力をオフ(ハイ・インピーダンス)にすることができないので，双方向ポートとしては使えないものでした．

ところが，PS/2用のパラレル・ポートには出力をオフ(ハイ・インピーダンス)にする機能が追加され，双方向ポートとして使えるようになりました．その後のAT互換機は，PC/AT互換の単方向パラレル・ポートを採用するものと，PS/2互換の双方向パラレル・ポートを採用するものに分かれました．ただし，単方向パラレル・ポートでも，**プリンタのステータス線**を利用して擬似的に双方向伝送を行うことは可能です．

双方向パラレル・ポートは単なるプリンタ・インターフェースだけでなく，汎用

セントロニクス互換プリンタ・インターフェース
アメリカのプリンタ・メーカのCentronics Data Computer社が，自社のプリンタ製品とコンピュータの接続のために定めた仕様．パソコンが登場するはるか前の1960年代に作られたものだが，後に多くのパソコン・メーカがプリンタ・インターフェースとして装備するようになった．PC/ATのパラレル・ポートはセントロニクスと互換性はあるが，一般的なセントロニクス仕様とはやや異なる独自の仕様だった．

単方向ポート
プリンタ・メーカだったCentronics社では，プリンタからパソコンにデータを送る必要はないと考えて，単方向の仕様を作ったものと思われる．

プリンタのステータス線
セントロニクスではプリンタの状態を示す信号線が何本か定義されている．それらはプリンタが出力するので，そのうち4本を利用してプリンタからパソコンに任意データを送る方法が工夫された．

インターフェースとしてパソコンどうしの接続や，パソコンと周辺機器の接続に使われるようになりました．たとえば，コストが高いSCSIのかわりに，外付けHDDなどの簡易インターフェースにしばしば利用されます．このような汎用の使い方が増えたことから，データ転送を高速化したいという要求が起こってきました．

そこで，従来の双方向パラレル・ポートの上位互換の高速転送方式として，Microsoft社などはECP（Extended Capabilities Port）を，Intel社などはEPP（Enchanced Parallel Port）を提唱しました．ECPやEPPを用いれば，最大2Mバイト/s（16Mビット/s）の高速双方向伝送が可能です．

規格化されたパラレル・ポート　IEEE 1284

このように，PC/AT以後のパラレル・ポートはさまざまに拡張が進められた結果，メーカによる仕様の違いが大きくなり，混乱を生じるようになりました．そこで，標準化団体の**IEEE**（アイトリプルイー）（Institute of Electrical and Electronics Engineers）においてパラレル・ポートの仕様の標準化が進められ，1994年にIEEE 1284-1994として承認されました．

IEEE
名称は電気電子技術者協会だが，とくにコンピュータのハードウェア，ソフトウェアの標準化を積極的に推進している．パソコン周辺規格では，CD-ROMやマルチメディア関係はISO/IEC，ハードディスク関連はANSI，電話やデータ通信はEIA/TIAとITU-T，システム・バスと周辺機器はIEEEというように住み分けができている．

IEEE 1284では，従来から存在するすべてのパラレル・ポートの仕様を整理し，5種類の動作モードを定義しています．PC/AT互換の単方向伝送を行う互換モード，単方向ポートで擬似的に双方向伝送を行うニブル・モード，PS/2互換の双方向伝送のバイト・モード，高速双方向伝送のECPモードとEPPモードの5種類です．各モードとも，これまで製品によってバラつきの大きかったタイミングや電気的仕様を規定し直しています．また，これまで統一されていなかったモード切り替えの手順も厳密に規定しています．

IEEE 1284準拠のパラレル・ポートは，この5種類のモードうち，互換モードはかならずサポートすることが必要です．それ以外の4種類のモードは，必要に応じて任意にサポートできます．

PC 95からPC 98へ

現在のAT互換機の多くはIEEE 1284準拠のパラレル・ポートを標準で装備しています．

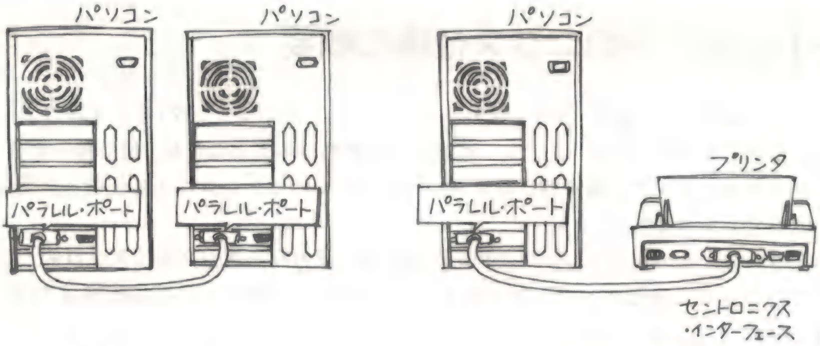
PC 95ハードウェアデザインガイドでは，ニブル・モード対応のIEEE 1284パラレル・ポートを必須条件とし，**ECP対応を推奨条件として**います．PC 97とPC 98では，新しい周辺インターフェースとしてUSBを採用したことから，パラレル・ポートは任意装備のデバイスとなりました．しかし，パラレル・ポートを装備する場合には，ニブル・モードとECPの両方に対応することを必須としています．

ECP対応を推奨条件として
高速双方向転送にはECPとEPPがあるが，もともとMicrosoft社が提唱したECPの方を普及させようとしているのだろう．

USBのフル・スピード・モード
USBには1.5Mビット/sのロー・スピード・モードと12Mビット/sのフル・スピード・モードがあって，周辺機器の種類によって使い分けられる．Microsoft社やIntel社などでは，それより高速のインターフェースは100/200/400ビット/sのIEEE 1394を普及させようと考えていたようだが，現状ではもう一つ普及にはずみがつかないようだ．

ニブル・モードを必須にしているのは，パラレル・ポートを双方向伝送に利用するためというより，PnP（プラグ&プレイ）のためにプリンタ側からデバイス情報を取り込むことが必要だからです．

PC 97やPC 98では，シリアル・ポート，パラレル・ポート，キーボード・コネクタ，マウス・コネクタなどの従来の周辺インターフェースをまとめて置き換えることをねらってUSBを採用しました．ただし，ECP/EPPモードのパラレル・ポートは最大16Mビット/s（2Mバイト/s）のデータ転送が可能であり，**USBのフル・スピード・モード**（12Mビット/s）よりも高速です．USBには一つのコ



《双方向パラレル・ポートはプリンタだけでなく，パソコンどうしも接続できる》

ネクタに多数の周辺機器を接続できる利点がありますが，周辺機器の台数が増えれば1台あたりの実効転送速度はさらに低下します．最大転送速度を240Mビット/sに高速化した新しいUSB規格（USB 2.0）も検討が進められていますが，実際に製品化されるのは2000年後半以降になると考えられます．

また，最近はUSB対接続のプリンタも広く出回ってきましたが，従来のパラレル・ポート接続のプリンタをすぐに廃棄するわけにはいきません．IEEE 1284はまだ当分の間使い続けられると思われます．

トランジスタ技術 エレクトロニクスの基礎と実用技術を満載したフィールド・ワーク・マガジン

SPECIAL No.65

B5判　160頁
定価 1,840円（税込）

特集　PCIバスの基礎と応用

Windows95パソコンの構成からCompactPCIシステムの構築まで

パソコン用OS Windowsの進化により，パソコンに求められるハードウェアも規格化されてきました．とくにPCIバスは拡張ボードの規格としてWindowsパソコンだけでなく，MacやAlpha PCにも採用されています．DOS/Vマシンで採用されたISAバス規格は，次世代のWindows98パソコンではお荷物になっています．
PCIというバス規格は，データ処理スピードの高速化に対応し，CPUとOSに依存しないように考えられたコンピュータ・システムの標準バスです．PCマザーボードや組み込み用のCPU基板を使ってコンピュータ・システムを組み上げる時の参考書としてお役立てください．

CQ出版社　〒170-8461　東京都豊島区巣鴨1-14-2　販売部　☎(03)5395-2141　振替 00100-7-10665

パラレル・ポートとセントロニクス仕様の概要

互換といっても
このパラレル・ポートに限らず，多くのパソコン・メーカでは セントロニクス仕様そのままを用いず，何らかの独自の変更を加えていることが多い．

PC/AT のパラレル・ポートはセントロニクス互換のプリンタ・インターフェース機能をもっています．ただし，**互換といっても**コネクタの形状，信号線の定義，信号のタイミング，電気的仕様など，オリジナルのセントロニクス仕様とは異なる部分もあります．

さらに，その後の PS/2 や AT 互換機では，双方向伝送や高速伝送を目的としてさまざまな拡張が行われてきました．現在では，それらの拡張仕様は IEEE 1284 によって統一されてきました．

ここでは，まずおおもとのセントロニクス仕様と，PC/AT のパラレル・ポートの仕様について解説します．

セントロニクス仕様という規格

自社のプリンタ製品
当時はメインフレームやミニコンピュータの時代であり，プリンタも大型のものが多かったが，Centronics 社では小型のドット・インパクト・プリンタを発売して好評を博した．やがて，パソコンが作られるようになると，手ごろな小型プリンタとしてパソコン用にも使われるようになった．

大量のデータを連続的に転送するようなことは
最近のプリンタは大容量のメモリを備えており，大量の印刷データをまとめて転送できるものが多い．

3 線ハンドシェイク
セントロニクス仕様の転送方式を説明するとき，この 3 線ハンドシェイクという呼び方がすっかり定着してしまった．だが，本来のセントロニクス仕様では，Busy は単にプリンタの状態を示すステータス信号であり，ホスト側からの/Strobe に対して/Ack で応答する 2 線ハンドシェイクのシステムである．

単なる印字処理
プリンタ側での 1 文字の印字処理は，ドット・ピンを射出してインク・リボンをたたき，印字ヘッドを 1 文字分移動させるだけの時間ですむ．

セントロニクス仕様はもともと 1960 年代中頃に Centronics Data Computer 社が**自社のプリンタ製品**のインターフェース仕様として定めたものです．それが，業界標準のプリンタ・インターフェースとして他メーカの製品にも広く採用されるようになりました．オリジナルの仕様書は Centronics 社から発行されていましたが，入手は難しいと思われます．

現在では，IEEE 1284 の規格書に付録（Annex C）としてセントロニクス仕様が収録されているので，容易に参照できます．これは，1980 年 4 月に発行された Centronics Engineering Standard Number 9, Revision B を収録したものです．Annex C には，PC/AT やその後の AT 互換機のパラレル・ポートの仕様も合わせて収録されています．

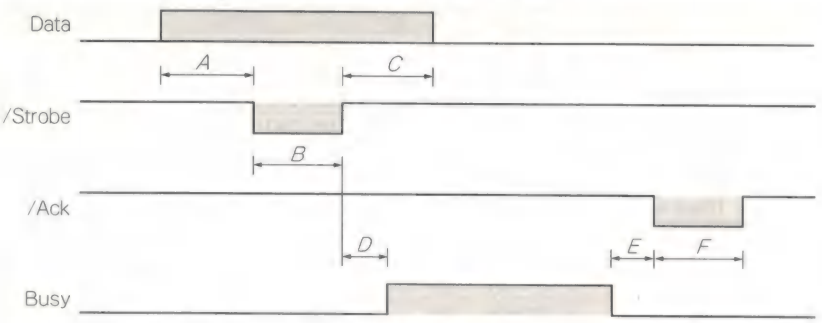
セントロニクス仕様は，まだパソコンが作られていない時代の古い規格であり，プリンタが 1 文字分ずつデータを受け取って機械的に印字を行うことを前提としています．**大量のデータを連続的に転送するようなことは**考えられていませんでした．

1 バイト（8 ビット）のデータをホスト・コンピュータからプリンタにパラレルに転送する単方向インターフェースです．このとき，ホスト側は/Strobe，プリンタ側は/Ack と Busy の制御線を使って，**3 線ハンドシェイク**方式で非同期データ転送を行います（**図 2-1**）．

/Strobe はデータ線上に有効データが出力されていることを示します．Busy はプリンタがデータ処理中（1 文字の印字中）で，次のデータを受け取れないことを示します．/Ack はプリンタのデータ受信が正常に終了したことを示します．ホストは/Strobe とともに 1 バイトのデータを出力します．プリンタは/Strobe がアクティブの期間内にデータを取り込みますが，その後も印字などデータ処理に必要な期間は Busy をアクティブにして，次のデータが送られてくるのを防ぎます．プリンタはデータ処理が終わったら/Ack をホストに返し，Busy を解除します．ホストは Busy と/Ack を確認して，次のデータの準備ができたらふたたび 1 バイトのデータを送ります．

このように，データを送るタイミングはホスト側とプリンタ側の双方の都合にあわせられます．機械式プリンタの場合，**単なる印字処理**と**キャリッジ・リターンなど**の制御コードの処理では処理時間が大きく異なります．セントロニクス仕様の 3 線ハンドシェイク方式は，プリンタの処理時間の違いに柔軟に対応できる

〈図 2-1〉セントロニクス仕様のハンドシェイク



(a)　タイミング図

記号	名称	min	max	単位
A	Dataセットアップ時間	1.0	—	μs
B	/Strobeパルス幅	1.0	500	μs
C	Dataホールド時間	1.0	—	μs
D	Busy遅延時間	0	1.5	μs
E	/Ack遅延時間	2.5	10	μs
F	/Ackパルス幅	2.5	6	μs

(b)　ハンドシェイクのタイミング

優れた方式でした．

最近では，ページ·プリンタのように1ページ分の印字データをまとめて受け取って，その後で印字処理を行うプリンタが多くなっています．Busy 時間を0としたときの1バイトの最小転送時間は，**図 2-1** から $A+B+D+E+F=7\ \mu s$ と規定されます．パソコンもプリンタも十分に高速なら，最大伝送速度は $1/7\ \mu s \fallingdotseq 143$ k バイト/s となります．実際はパソコンの処理時間もあり，**せいぜい 100 k バイト/s 程度**と言われています．

データ線とハンドシェイク線のほかに，ホスト側からプリンタをリセットするための制御線（/Input Prime）や，プリンタの状態をホストに伝えるためのいくつかのステータス線があります（**表 2-1**）．ただし，信号線の定義や使い方はかならずしも統一されていません．もともとプリンタ·メーカの Centronics Data Computer 社が作った規格なので，新しい機能を追加した新製品を出すときに，少しずつ仕様が拡張されてきたという経緯があります．さらに，セントロニクス·インターフェースを採用したパソコン·メーカが，それぞれ**独自の解釈で**機能を変更や省略してしまうことも多いからです．

3線ハンドシェイクにしても，もともと Busy は単にプリンタの処理中を意味するだけで，正常終了か異常終了かは/Ack と/Fault のどちらを返すかで決めていました．しかし，パソコンによっては/Ack や/Fault を監視せず，Busy の終了だけを見て次のデータを送る動作をするものもあります．

コネクタは 36 ピンの**アンフェノール·コネクタ**です（**図 2-2**）．信号が TTL レベルでノイズに弱いため，主要な信号線には個別リターン線を設けて，ツイスト·ペア線で伝送します．これもパソコンによっては，信号線やリターン線の数を減らして 25 ピンや 14 ピンの小型コネクタを採用する**互換インターフェース**が多くなっています．

セントロニクスの電気的仕様は TTL レベルですが，ドライバの出力電流はソ

キャリッジ・リターンなど

最大でラインの右端まで移動した印字ヘッドを，ラインの左端まで移動する（CR）のに印字の何倍も時間がかかる．さらに，ペーパー・ドラムを回転させて用紙を1行分移動させる（LF）のにもかなりの時間がかかる．

せいぜい 100 k バイト/s 程度

それでもシリアル・ポートよりずっと高速であり「これで双方向だったらいろいろ使えるのに…」と多くのユーザを嘆かせてきた．

独自の解釈で

そのときの都合に合わせて，好き勝手に変更してしまうことが多いようだ．たとえば，NEC などいくつかのメーカーでは，パソコンのバック・ボードに 36 ピン・コネクタを搭載するスペースがなかったため，信号のリターンを省略して 14 ピン・セントロニクス・コネクタを採用した．IBM 社でも同様の理由から，25 ピン D-Sub コネクタを採用した．その他，信号線の種類やタイミング，ハンドシェイクの方法もしばしば勝手に変更されている．

アンフェノール・コネクタ

アメリカの Amphenol 社が製品化したインターフェース用コネクタ．コンピュータ周辺で広く採用され，デファクト・スタンダードといえる．2.54 mm ピッチ，角形，多接点で，D-sub コネクタと競合する．D-sub がジグザグ配列のピン接点なのに対して，Amphenol は対向配列のリボン接点である．脱着が容易でピン折れなどの心配が少ないため実用的には使いやすいが，D-sub が ISO 規格になっているのに対して，Amphenol は公的には規格化されていない．最近は D-sub，アンフェノールとも次第にハーフ・ピッチのものに主流が移行しているようだ．

互換インターフェース

パソコン側のインターフェースに独自の解釈を導入しても，変換ケーブルを用いるなどの方法でセントロニクス仕様のプリンタと「かろうじて」接続可能であれば，互換と称していました．

〈表 2-1〉 セントロニクス仕様のコネクタのピン接続

ピン番号	信号名	方向	ピン番号	信号名	方向
1	/Strobe	H→P	19	/Strobe のリターン	−
2	Data 1	H→P	20	Data 1 のリターン	−
3	Data 2	H→P	21	Data 2 のリターン	−
4	Data 3	H→P	22	Data 3 のリターン	−
5	Data 4	H→P	23	Data 4 のリターン	−
6	Data 5	H→P	24	Data 5 のリターン	−
7	Data 6	H→P	25	Data 6 のリターン	−
8	Data 7	H→P	26	Data 7 のリターン	−
9	Data 8	H→P	27	Data 8 のリターン	−
10	/Ack	P→H	28	/Ack のリターン	−
11	Busy	P→H	29	Busy のリターン	−
12	Paper Empty	P→H	30	/Input Prime のリターン	−
13	Select	P→H	31	/Input Prime	H→P
14	Signal Ground	−	32	/Fault	P→H
15	OscExt	P→H	33	Light Detect	P→H
16	Signal Ground	−	34	Line Count	P→H
17	Chassis Ground	−	35	Line Count Return	P→H
18	+5V	P→H	36	予約	P→H

信号名の前の/は負論理（ロー・アクティブ）を示す．
H：ホスト・コンピュータ
P：プリンタ
データ・ビットの並び順は任意．一般的には Data 1=LSB，Data 8=MSB．

ース 320 μA，シンク 14 mA が必要です（**表 2-2**）．LS のゲート IC ではこれを満たすことはできず，LS240 などのバッファが用いられます．

トランジスタ技術 SPECIAL No.61
エレクトロニクスの基礎と実用技術を満載したフィールド・ワーク・マガジン
B5 判　176 頁
定価 1,840 円（税込）

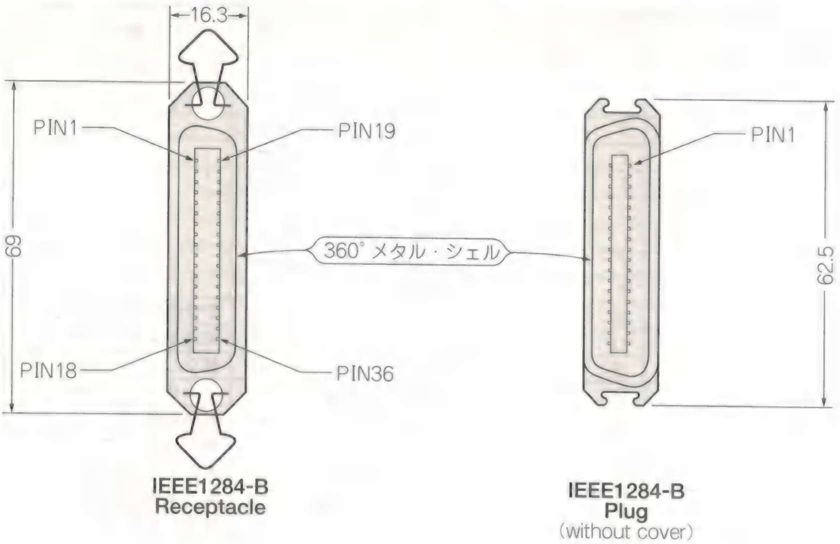
特集　モータ制御 & メカトロ技術入門
いろいろなモータとその駆動法を理解しよう

身の回りには数多くのモータが使われています．もちろんモータ単体ではなく，モータを組み込んだ機器がゴマンとあります．今回は，モータの種類とその動作原理をわかりやすく解説します．モータの駆動回路や制御技術，メカトロニクスとそれを構成する機器を紹介し，実際の例をくわしく解説します．

第 1 章　キー・テクノロジーとしてのモータ技術
コラム　私とモータの出会い
第 2 章　メカトロニクスとそれを構成する機器
第 3 章　モータの種類と動作原理
第 4 章　DCモータの駆動方法とその動作
第 5 章　ミニ四駆で学ぶモータ技術
コラム　ミニ四駆の歴史
第 6 章　モータ計測システムの製作
第 7 章　ステッピング・モータの制御法
第 8 章 モータ制御基板の設計

CQ出版社　〒170-8461　東京都豊島区巣鴨 1-14-2　販売部　☎ (03) 5395-2141　振替 00100-7-10665

〈図 2-2〉セントロニクス仕様のコネクタ（単位；mm）



〈表 2-2〉セントロニクス仕様インターフェースの電気的特性

項目	記事	min	max	単位
信号電圧	絶対定格	-0.5	+5.5	V
	推奨範囲	0	+5	V
	論理‘1’	+2.4	+5	V
	論理‘0’	0	+0.4	V
電流容量	論理‘1’，電圧 2.4V	− 320	−	μ A
	論理‘0’，電圧 0.4V	14	−	mA

PC/AT のパラレル・ポート

PC/AT のパラレル・ポートは，オリジナルのセントロニクス仕様よりもプリンタが出力するステータス線を減らし，そのかわりホストが出力する制御線を増やしています．新たに追加された制御線は，自動給紙（/AutoFdXT）とプリンタ選択（/Select In）です（**表 2-3**）．

また，**リターン線を二つずつまとめる**ことによって全体の線の本数を減らし，25 ピン D-sub コネクタに収容しています（**図 2-3**）．セントロニクス仕様の 36 ピン・アンフェノールは PC/AT の拡張ボードには大きすぎて，複合機能ボードでほかの機能のコネクタを装着する余地がなくなってしまったためです．なお，25 ピン D-sub は一般のパソコンでは RS-232-C ポートに用いられているため，RS-232-C ポートとはコネクタのオス・メスを逆にしてケーブルの誤接続を防いでいます．RS-232-C ポートは本体側がオスなのに対して，PC/AT のパラレル・ポートは本体側がメスとなっています．

リターン線を二つずつまとめる
プリンタ側コネクタでは /Strobe，Data1 ～ Data8，/Ack のそれぞれに対してリターン線が定義されているが，ホスト側では /Strobe と Data1，Data2 と Data3，…，Data8 と /Ack というように 2 本ずつまとめて 1 本のピンに接続している．

基本的な 3 線ハンドシェイクはオリジナルのセントロニクス仕様と同じですが，タイミングはやや異なっています（**図 2-4**）．とくに，Data のセットアップ/ホールド時間，/Strobe のパルス幅の最小値がいずれも 0.5 μs に短縮されました．これによって，Busy 時間を 0 としたときの 1 バイトの最小転送時間は，**図 2-4** から $A+B+D+E$=3.5 μs と規定されます．規定から許される伝送速度の上

〈表 2-3〉PC/AT 仕様のパラレル・ポートのピン接続

ピン番号		信号名	方向	備考
プリンタ (36ピン)	ホスト (25ピン)			
1	1	/Strobe	H→P	有効データ出力中
2	2	Data 1	H→P	印字データ
3	3	Data 2	H→P	
4	4	Data 3	H→P	
5	5	Data 4	H→P	
6	6	Data 5	H→P	
7	7	Data 6	H→P	
8	8	Data 7	H→P	
9	9	Data 8	H→P	
10	10	/Ack	P→H	正常終了ステータス
11	11	Busy	P→H	処理中ステータス
12	12	Paper Empty	P→H	紙なしステータス
13	13	Select (On-line)	P→H	オンライン・ステータス
14	14	/AutoFdXT	H→P	自動給紙 (*)
15	−	−		使用しない (*)
16	−	Signal Ground		ホスト側は接続しない
17	−	Chassis Ground		ホスト側は接続しない
18	−	−		使用しない (*)
19	18	/Strobe のリターン		
20		Data 1 のリターン		
21	19	Data 2 のリターン		
22		Data 3 のリターン		
23	20	Data 4 のリターン		
24		Data 5 のリターン		
25	21	Data 6 のリターン		
26		Data 7 のリターン		
27	22	Data 8 のリターン		
28		/Ack のリターン		
29	24	Busy のリターン		
30	25	/Init のリターン		
31	16	/Init (/Reset)	H→P	/Input Prime に相当
32	15	/Fault	P→H	異常発生ステータス
33	23	/Select In のリターン		(*)
34	−	−		使用しない (*)
35	−	+5V	P→H	ホスト側は接続しない
36	17	/Select In	H→P	プリンタ選択 (*)

信号名の前の/は負論理 (ロー・アクティブ) を示す.
H： ホスト・コンピュータ
P：プリンタまたはペリフェラル
(＊) 印はセントロニクス仕様と定義が異なるピン

限は 1/3.5 μs ≒ 286 k バイト/s となります.

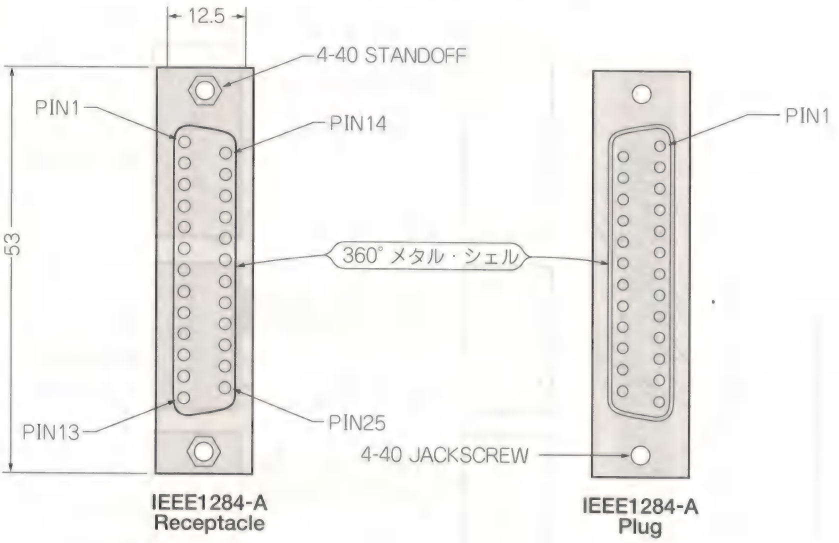
また，セントロニクス仕様では Busy が終了した後に出力されることになっていた/Ack パルスが，Busy のアクティブ期間に出力されてもよいことになりました．さらに，Busy 遅延時間と/Ack パルス幅の上限もなくなりました．これは何を意味しているかというと，PC/AT のパラレル･ポートでは，**/Ack は単に Busy の反転でもかまわない**ということです．もともとのセントロニクス仕様のように Busy と/Ack を使い分けず，Busy パルスの後エッジと/Ack パルスの後エッジを確認したら次のデータを送ります.

電気的仕様は TTL レベルですが，データ線のドライバの出力電流はソース 2.6 mA，シンク 24 mA が必要です (**表 2-4**)．LS のゲート IC ではこれを満たすことはできず，LS240 などのバッファが用いられます．また，制御線のドライバはオープン･コレクタ (4.7 kΩでプルアップ) となっています.

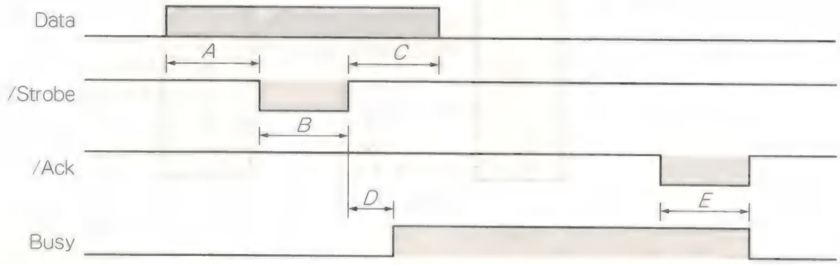
/Ack は単に Busy の反転でもかまわない

パラレル・ポートは，パソコン (PC/AT) 側では/Ack と Busy を使い分けずに処理を簡単にすませるから，プリンタ側でも手を抜いて/Ack と Busy を同じ信号として (極性だけ反転して) 出力すればいい，という仕様になっている.

〈図 2-3〉 AC/AT 仕様のパラレル・ポートのコネクタ（IEEE1284-A）



〈図 2-4〉 PC/AT 仕様のパラレル・ポートのハンドシェイク



(a) プリンタ・インターフェースのタイミング図

記号	名称	min	max	単位
A	Dataセットアップ時間	0.5	−	μs
B	/Strobeパルス幅	0.5	500	μs
C	Dataホールド時間	0.5	−	μs
D	Busy遅延時間	0	−	μs
E	/Ackパルス幅	2.5	−	μs

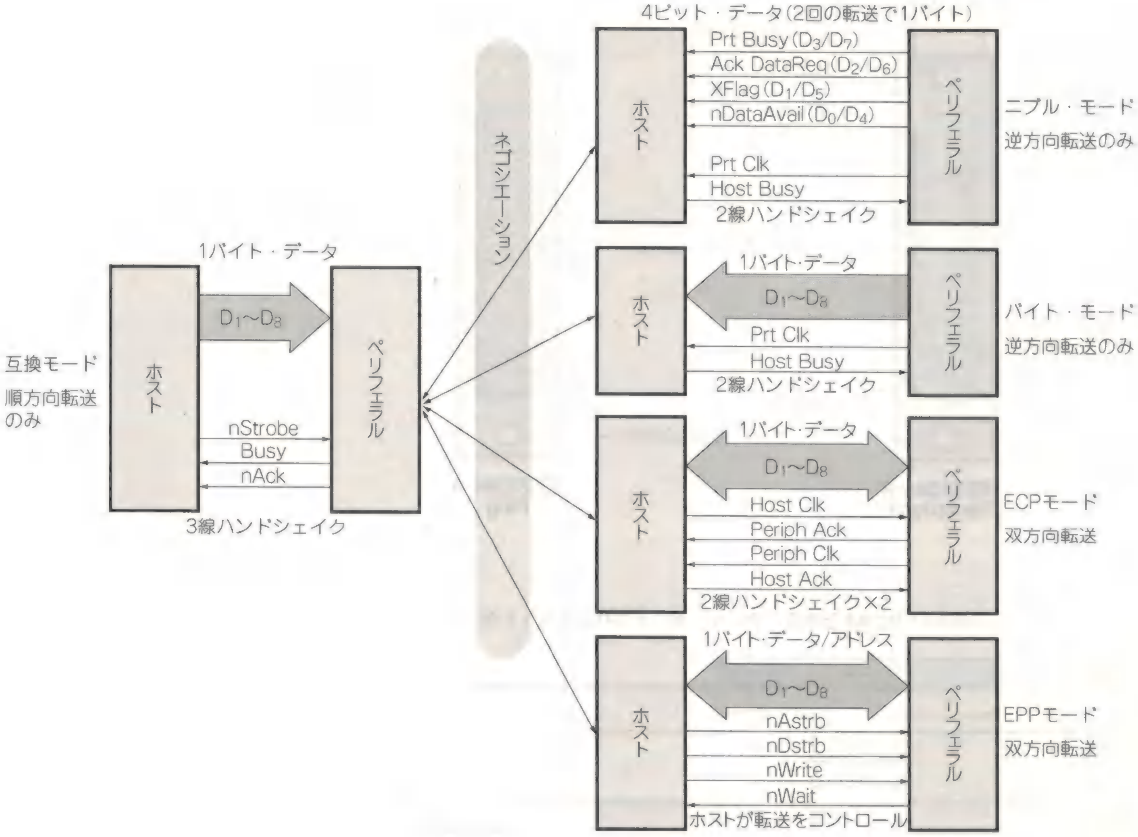
(b) ハンドシェイクのタイミング

〈表 2-4〉 PC/AT 仕様のパラレル・ポートの電気的仕様

項目	記事		min	単位
信号電圧	74LS ファミリに従う			
電流容量	制御線	論理 '1', 2.4V	−	−
	(オープン・コレクタ)	論理 '0', 0.4V	16	mA
	データ線	論理 '1', 2.4V	−2.6	mA
	(LS のバッファ・タイプ)	論理 '0', 0.4V	24	mA

制御線は/Strobe, /Init, /AutoFdXT, /Select In
制御線は 4.7k Ω で 5V にプルアップ

〈図 2-5〉IEEE1284 の互換モード



IEEE 1284 規格の概要

それ以上の機能はすべてオプション

IEEE 1284 では，単方向伝送の互換モードだけを必須としている．データをプリンタに印字するだけの目的なら，単方向伝送で十分だからだ．Microsoft 社の PC97, PC98 では，互換モードに加えてニブル・モードと ECP モードの二つの双方向伝送を必須としている．これは，PnP (プラグ&プレイ) を実現するためにプリンタからの情報を取得することが必要だからだ．

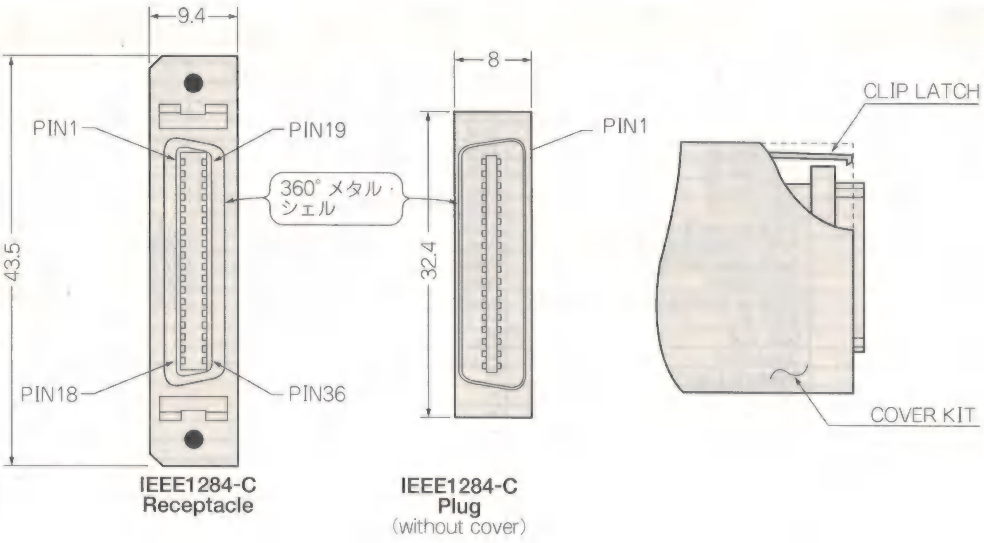
AT 互換機のパラレル･ポートは，メーカによって仕様に細かい違いがあり，また双方向伝送や高速伝送をめざして独自の拡張が進められてきたため，機種間の互換性に問題が生じるようになってきました．IEEE 1284 は，パラレル･ポートの標準仕様を定めることによって，異機種間の相互接続性を改善することを目的として作られました．

IEEE 1284 では，従来のパラレル･ポート相当の単方向伝送を互換モードと呼び，拡張機能としてニブル･モード，バイト･モード，ECP モード，EPP モードの四つの仕様を規定しています (**図 2-5**)．互換モードは必須の機能であり，**それ以上の機能はすべてオプション**です．オプションのモードを利用するには，ホスト (パソコン) とペリフェラル (周辺機器) がネゴシエーションを実行して，合意が成立したら伝送を実行できます．

IEEE 1284 のピン接続

IEEE 1284 はホスト (パソコン) とペリフェラル (周辺機器) の 1 対 1 インターフェースとして定義されています (**表 2-5**)．プリンタだけでなく，さまざま

〈図 2-6〉IEEE1284 の C コネクタ（単位；mm）



な周辺機器を接続することを考慮して信号名が決められています．もともとのセントロニクス仕様ではプリンタの制御用やステータス用だった信号を，**汎用の伝送制御用の信号**に転用しています．

なお，セントロニクス仕様と PC/AT 仕様では負論理の信号名に“/”を付けて区別していましたが，IEEE 1284 仕様では負論理の信号名に“n”を付けて区別するように変わりました．たとえば，セントロニクス仕様と PC/AT 仕様の“/Ack”は，IEEE 1284 仕様では“nAck”になります．

IEEE 1284 では互換モード，ニブル・モード，バイト・モード，ECP モード，EPP モードの 5 種類の動作モードを定義しており，モードによって信号名が変わります．ホスト側，ペリフェラル側とも，パワー・オン後はまず互換モードで動作します．互換モードのピン接続は，PC/AT の単方向パラレル・ポートと同じです．その後，双方向伝送や高速伝送の必要があれば，規定された手順でネゴシエーションを行ってほかのモードに移行し，信号の定義が変わります．

IEEE 1284 ではホスト側の 25 ピン D-sub を 1284-A コネクタ，ペリフェラル側の 36 ピン・アンフェノールを 1284-B コネクタと呼んでいます．さらに，もう一つ新しいコネクタとして 36 ピン・**ハーフピッチ・アンフェノール**を採用し，これを 1284-C コネクタと呼んでいます．1284-C コネクタはホスト側，ペリフェラル側ともに用いられます（**図 2-6**）．

なお，ピン数は同じ 36 ピンですが，1284-C コネクタは従来の 1284-B コネクタとはピン接続が異なるので注意が必要です．

IEEE 1284 で新しく追加された信号として，Peripheral Logic High と Host Logic High があります．Peripheral Logic High はペリフェラルが出力し，ペリフェラルの動作が有効であることを示します．ホストは，この信号を監視してペリフェラルの状態を知ることができます．ただし，1284-A コネクタでは Peripheral Logic High は省略されているため，25 ピン D-sub のホストはこの信号を利用できません．

Host Logic High はホストが出力し，ホストの動作が有効であることを示します．ペリフェラルは，この信号を監視してホストの状態を知ることができます．

汎用の伝送制御用の信号
もちろん，相手がプリンタの場合は従来のプリンタ制御/ステータス用の信号として用いる．そのため，IEEE 1284 の多くの信号線は，モードごとに名称や機能がまったく変わってしまうことがあるので注意が必要だ．

ハーフピッチ・アンフェノール
接点間隔が 2.54 mm ピッチの従来のアンフェノール（フルピッチ）に対して，それを 1.27 mm に縮小して小型化したコネクタ．パソコンや周辺機器の小型化に合わせて，コネクタも小型化している．

〈表 2-5〉IEEE1284 の信号線

ピン番号		信号名	互換モード		ニブル・モード		
Bコネクタ(36ピン)	Aコネクタ(25ピン)	( ) 内は互換モードの信号名	機能	方向	機能	逆方向転送時	方向
1	1	HostClk/nWrite (nStrobe)	nStrobe	H → P	HostClk	Hi に保持	H → P
2	2	AD1 (Data 1)	Data 1 (LSB)	H → P	Data 1 (LSB)	–	–
3	3	AD2 (Data 2)	Data 2	H → P	Data 2	–	–
4	4	AD3 (Data 3)	Data 3	H → P	Data 3	–	–
5	5	AD4 (Data 4)	Data 4	H → P	Data 4	–	–
6	6	AD5 (Data 5)	Data 5	H → P	Data 5	–	–
7	7	AD6 (Data 6)	Data 6	H → P	Data 6	–	–
8	8	AD7 (Data 7)	Data 7	H → P	Data 7	–	–
9	9	AD8 (Data 8)	Data 8 (MSB)	H → P	Data 8 (MSB)	–	–
10	10	PtrClk/PeriphClk/Intr (nAck)	nAck	P → H	PtrClk	ストローブ	P → H
11	11	PtrBusy/PeriphAck/nWait (Busy)	Busy	P → H	PtrBusy	Data 3/7	P → H
12	12	AckDataReq/nAckReverse (PError)	PError	P → H	AckDataReq	Data 2/6	P → H
13	13	Xflag (Select)	Select	P → H	Xflag	Data 1/5	P → H
14	14	HostBusy/HostAck/nDStrb (nAutoFd)	nAutoFd	H → P	HostBusy	ビジー	H → P
15	–	使用しない	←		←		
16	–	Logic Gnd	←		←		
17	–	Chassis Gnd	←		←		
18	–	Peripheral Logic High	←		←		
19	18	nStrobe のリターン	←		←		
20	19	Data 1 のリターン	←		←		
21		Data 2 のリターン	←		←		
22	20	Data 3 のリターン	←		←		
23		Data 4 のリターン	←		←		
24	21	Data 5 のリターン	←		←		
25		Data 6 のリターン	←		←		
26	22	Data 7 のリターン	←		←		
27		Data 8 のリターン	←		←		
28	24	PError, Select, nAck のリターン	←		←		
29	23	Busy と nFault のリターン	←		←		
30	25	nAutoFd, nSelectIn, nInit のリターン	←		←		
31	16	nReverseRequest (nInit)	nInit	H → P	nInit	Hi に保持	H → P
32	15	nDataAvail/nPeriphRequest (nFault)	nFault	P → H	nDataAvail	Data 0/4	P → H
33	–	使用しない	←		←		
34	–	使用しない	←		←		
35	–	使用しない	←		←		
36	17	1284Active/nAStrb (nSelectIn)	nSelectIn	H → P	1284Active	Hi に保持	H → P

信号名の前の n は負論理 (ローアクティブ) を示す.
H：ホスト　P：プリンタまたはペリフェラル

(a) ピン接続 (1284-A コネクタ/B コネクタ)

ピン番号	信号名	互換モード		ニブル・モード		
Cコネクタ(36ピン)	( ) 内は互換モードの信号名	機能	方向	機能	逆方向転送時	方向
1	PtrBusy/PeriphAck/nWait (Busy)	Busy	P → H	PtrBusy	Data 3/7	P → H
2	Xflag (Select)	Select	P → H	Xflag	Data 1/5	P → H
3	PtrClk/PeriphClk/Intr (nAck)	nAck	P → H	PtrClk	ストローブ	P → H
4	nDataAvail/nPeriphRequest (nFault)	nFault	P → H	nDataAvail	Data 0/4	P → H
5	AckDataReq/nAckReverse (PError)	PError	P → H	AckDataReq	Data 2/6	P → H
6	AD1 (Data 1)	Data 1 (LSB)	H → P	Data 1 (LSB)	–	–
7	AD2 (Data 2)	Data 2	H → P	Data 2	–	–
8	AD3 (Data 3)	Data 3	H → P	Data 3	–	–
9	AD4 (Data 4)	Data 4	H → P	Data 4	–	–
10	AD5 (Data 5)	Data 5	H → P	Data 5	–	–
11	AD6 (Data 6)	Data 6	H → P	Data 6	–	–
12	AD7 (Data 7)	Data 7	H → P	Data 7	–	–
13	AD8 (Data 8)	Data 8 (MSB)	H → P	Data 8 (MSB)	–	–
14	nReverseRequest (nInit)	nInit	H → P	nInit	Hi に保持	H → P
15	HostClk/nWrite (nStrobe)	nStrobe	H → P	HostClk	Hi に保持	H → P
16	1284Active/nAStrb (nSelectIn)	nSelectIn	H → P	1284Active	Hi に保持	H → P
17	HostBusy/HostAck/nDStrb (nAutoFd)	nAutoFd	H → P	HostBusy	ビジー	H → P
18	Host Logic High	←		←		
19	Busy のリターン	←		←		
20	Select のリターン	←		←		
21	nAck のリターン	←		←		
22	nFault のリターン	←		←		
23	PError のリターン	←		←		
24	Data 1 のリターン	←		←		
25	Data 2 のリターン	←		←		
26	Data 3 のリターン	←		←		
27	Data 4 のリターン	←		←		
28	Data 5 のリターン	←		←		
29	Data 6 のリターン	←		←		
30	Data 7 のリターン	←		←		
31	Data 8 のリターン	←		←		
32	nInit のリターン	←		←		
33	nStrobe のリターン	←		←		
34	nSelectIn のリターン	←		←		
35	nAutoFd のリターン	←		←		
36	Peripheral Logic High	←		←		

信号名の前の n は負論理 (ローアクティブ) を示す.
H：ホスト　P：プリンタまたはペリフェラル

(b) ピン接続 (1284-C コネクタ)

バイトモード			ECP モード				EPP モード			
機能	逆方向転送時	方向	機能	順方向転送時	逆方向転送時	方向	機能	順方向転送時	逆方向転送時	方向
HostClk	−	H → P	HostClk	ストローブ	−	H → P	nWrite	ライト	Hi に保持	H → P
Data 1 (LSB)	データ	P → H	Data 1 (LSB)	データ	データ	H → P/P → H	AD1	アドレス/データ	アドレス/データ	H → P/P → H
Data 2	データ	P → H	Data 2	データ	データ	H → P/P → H	AD2	アドレス/データ	アドレス/データ	H → P/P → H
Data 3	データ	P → H	Data 3	データ	データ	H → P/P → H	AD3	アドレス/データ	アドレス/データ	H → P/P → H
Data 4	データ	P → H	Data 4	データ	データ	H → P/P → H	AD4	アドレス/データ	アドレス/データ	H → P/P → H
Data 5	データ	P → H	Data 5	データ	データ	H → P/P → H	AD5	アドレス/データ	アドレス/データ	H → P/P → H
Data 6	データ	P → H	Data 6	データ	データ	H → P/P → H	AD6	アドレス/データ	アドレス/データ	H → P/P → H
Data 7	データ	P → H	Data 7	データ	データ	H → P/P → H	AD7	アドレス/データ	アドレス/データ	H → P/P → H
Data 8 (MSB)	データ	P → H	Data 8 (MSB)	データ	データ	H → P/P → H	AD8	アドレス/データ	アドレス/データ	H → P/P → H
PtrClk	ストローブ	P → H	PeriphClk	−	ストローブ	P → H	Intr	割り込み	割り込み	P → H
PtrBusy	−	P → H	PeriphAck	アクノリッジ	コマンド転送	P → H	nWait	ウェイト	ウェイト	P → H
AckDataReq	転送データあり	P → H	nAckReverse	逆方向受入	逆方向受入	P → H	User defined 1	−	−	P → H
Xflag	Hi に保持	P → H	Xflag	Hi に保持	Hi に保持	P → H	User defined 3	−	−	P → H
HostBusy	アクノリッジ	H → P	HostAck	コマンド転送	アクノリッジ	H → P	nDStrb	データ転送	データ転送	H → P
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
nInit	Hi に保持	H → P	nReverseRequest	逆方向要求	逆方向要求	H → P	nInit	EPP モード終了	EPP モード終了	H → P
nDataAvail	転送データあり	P → H	nPeriphReqest	転送データあり	転送データあり	P → H	User defined 2	−	−	P → H
←			←				←			
←			←				←			
←			←				←			
1284Active	Hi に保持	H → P	1284Active	Hi に保持	Hi に保持	H → P	nAStrb	アドレス転送	アドレス転送	H → P

バイトモード			ECP モード				EPP モード			
機能	逆方向転送時	方向	機能	順方向転送時	逆方向転送時	方向	機能	順方向転送時	逆方向転送時	方向
PtrBusy	−	P → H	PeriphAck	アクノリッジ	コマンド転送	P → H	nWait	ウェイト	ウェイト	P → H
Xflag	Hi に保持	P → H	Xflag	Hi に保持	Hi に保持	P → H	User defined 3	−	−	P → H
PtrClk	ストローブ	P → H	PeriphClk	−	ストローブ	P → H	Intr	割り込み	割り込み	P → H
nDataAvail	転送データあり	P → H	nPeriphReqest	転送データあり	転送データあり	P → H	User defined 2	−	−	P → H
AckDataReq	転送データあり	P → H	nAckReverse	逆方向受入	逆方向受入	P → H	User defined 1	−	−	P → H
Data 1 (LSB)	データ	P → H	Data 1 (LSB)	データ	データ	H → P/P → H	AD1	アドレス/データ	アドレス/データ	H → P/P → H
Data 2	データ	P → H	Data 2	データ	データ	H → P/P → H	AD2	アドレス/データ	アドレス/データ	H → P/P → H
Data 3	データ	P → H	Data 3	データ	データ	H → P/P → H	AD3	アドレス/データ	アドレス/データ	H → P/P → H
Data 4	データ	P → H	Data 4	データ	データ	H → P/P → H	AD4	アドレス/データ	アドレス/データ	H → P/P → H
Data 5	データ	P → H	Data 5	データ	データ	H → P/P → H	AD5	アドレス/データ	アドレス/データ	H → P/P → H
Data 6	データ	P → H	Data 6	データ	データ	H → P/P → H	AD6	アドレス/データ	アドレス/データ	H → P/P → H
Data 7	データ	P → H	Data 7	データ	データ	H → P/P → H	AD7	アドレス/データ	アドレス/データ	H → P/P → H
Data 8 (MSB)	データ	P → H	Data 8 (MSB)	データ	データ	H → P/P → H	AD8	アドレス/データ	アドレス/データ	H → P/P → H
nInit	Hi に保持	H → P	nReverseRequest	逆方向要求	逆方向要求	H → P	nInit	EPP モード終了	EPP モード終了	H → P
HostClk	−	H → P	HostClk	ストローブ	−	H → P	nWrite	ライト	Hi に保持	H → P
1284Active	Hi に保持	H → P	1284Active	Hi に保持	Hi に保持	H → P	nAStrb	アドレス転送	アドレス転送	H → P
HostBusy	アクノリッジ	H → P	HostAck	コマンド転送	アクノリッジ	H → P	nDStrb	データ転送	データ転送	H → P
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			
←			←				←			

ただし，1284-A および 1284-B コネクタでは Host Logic High は省略されています．1284-C コネクタの機器どうしを接続した場合だけ利用できます．

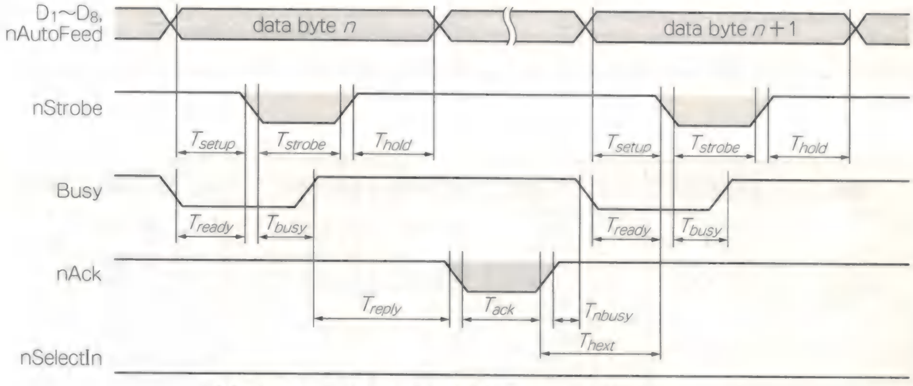
互換モードのハンドシェイク

互換モードは IBM の単方向パラレル・ポートとほぼ同じもので，IEEE 1284 の**デフォルトのモード**です．nStrobe，Busy，nAck を使った 3 線ハンドシェイクを行います．タイミングは PC/AT のパラレル・ポートよりも詳細に規定されていますが，基本的には従来と変わっていません（**図 2-7**）．

とくに変化した点としては，Busy パルスと nAck パルスの出始めのタイミング，nAck パルス幅の二つがあります．従来は，Busy パルスと nAck パルスは nStrobe パルスの後エッジの後に出力されることになっていましたが，実際のプリンタには nStrobe の後エッジより前に Busy や nAck が出始めるものもありました．IEEE 1284 では，Busy パルスは nStrobe の前エッジから 0.5 μs 以内に出力するように定めており，nAck も nStrobe の後エッジより前に出力できます．また，従来は nAck パルス幅の最小値は 2.5 μs と規定されていましたが，それを 0.5 μs に短縮しました．ペリフェラルの応答が十分に高速であれば，

デフォルトのモード
ホスト側，ペリフェラル側とも電源投入後はかならず互換モードになり，他のモードに移行するには定められたネゴシエーションの手順を実行しなければならない．

〈図 2-7〉互換モードのハンドシェイク



(a) 互換モードのハンドシェイクのタイミング図

記号	略号	名称	min	max	単位
A	T_{setup} (host)	Dataセットアップ時間(ホスト出力)	0.75	−	μs
	T_{setup} (peripheral)	Dataセットアップ時間(ペリフェラル入力)	−	0.5	μs
B	T_{strobe} (host)	/Strobeパルス幅(ホスト出力)	0.75	500	μs
	T_{strobe} (peripheral)	/Strobeパルス幅(ペリフェラル入力)	−	0.5	μs
C	T_{hold} (host)	Dataホールド時間(ホスト出力)	0.75	−	μs
	T_{hold} (peripheral)	Dataホールド時間(ペリフェラル入力)	−	0.5	μs
D	T_{busy}	Busy遅延時間(ペリフェラル出力)	−	0.5	μs
E	−	/Ack遅延時間	−	−	μs
F	T_{ack}	/Ackパルス幅(ペリフェラル出力)	0.5	10	μs
−	T_{ready}	Busy-/Strobeセットアップ時間(ホスト出力)	0	−	μs
−	T_{reply}	/Ack応答時間(ペリフェラル出力)	0	−	μs
−	T_{nbusy}	Busy終了時間(ペリフェラル出力)	0	−	μs
−	T_{next}	/Ack-/Strobeセットアップ時間(ホスト出力)	0	−	μs

(b) 互換モードのタイミング

nStrobeのパルスの期間内にBusyと nAckによる応答を終了してしまうことも可能です．

これによって，Busy時間を0としたときの1バイトの最小転送時間は，**図2-7**から$T_{setup}+T_{strobe}+T_{hold}$=2.25 μsと規定されます．規定から許される伝送速度の上限は1/2.25 μs≒444kバイト/sとなります．

互換モードではホスト→ペリフェラルの単方向の伝送しかできません．ホストとペリフェラルの両方がほかの四つのモードのどれかをサポートしている場合，ネゴシエーションを行って，成功すればそのモードに移行します．

ニブル・モードとバイト・モードは，互換モードに双方向伝送の機能を付加するためのモードです．順方向の伝送は互換モードで行い，逆方向の伝送を行う場合だけ，ニブル・モードやバイト・モードに移行します．

ECPモードとEPPモードは，より高速の双方向伝送を行うためのモードです．いったんネゴシエーションを行ってECPモードやEPPモードに移行したら，順方向も逆方向もそのモードで伝送を実行できます．もちろん，互換モードに戻って順方向の伝送を行うことも可能です．また，ECPモードにはRLE（Run Length Encoding）方式のデータ圧縮を行うモードがあります．

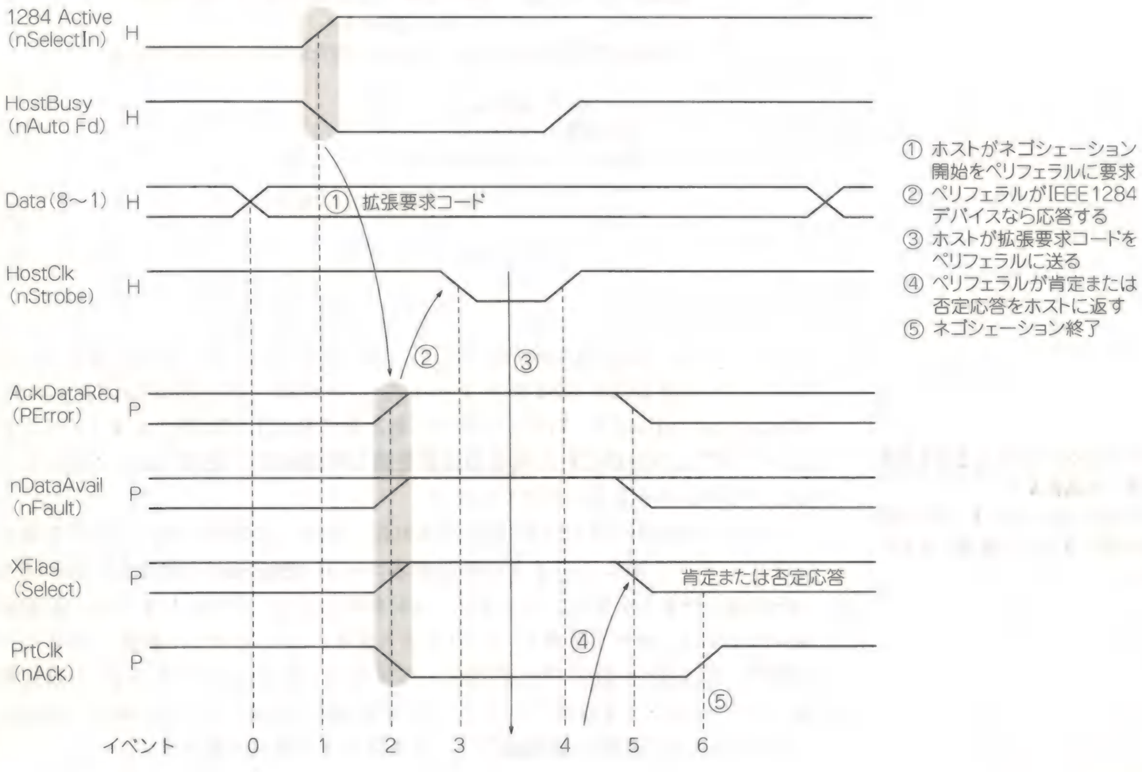
nStrobeのパルスの期間内に
旧タイプのプリンタと接続しても正常動作ができるように，nStrobeのパルス幅はあまり狭くすることができない．もともとのセントロニクス仕様では最小1μs, PC/ATのパラレル・ポートでは最小0.5μ，IEEE 1284では間をとって最小0.75μsとしている．これ以上狭くすると，プリンタによってはnStrobeのパルスを見落とす危険があるためだ．したがって，この制約の中で伝送を最大限に高速化するには，ペリフェラル側の処理をnStrobeのパルスの期間内に終わらせてしまうことが必要だ．

ネゴシエーションをしてモード変更する

ネゴシエーションでは，ホストが移行したいモードをペリフェラルに示し，ペリフェラルが受け入れの応答をすれば，そのモードに移行します（**図2-8**）．

ホストは，まずネゴシエーション開始の要求として，1284 Active（nSelection）

〈図2-8〉ネゴシエーションの手順（Hはホストが出力，Pはペリフェラルが出力）



〈表 2-6〉 互換モードの電気的仕様

コード (16 進)	コード (2 進)	ビット (注)	定義		肯定応答
0x80	10000000	b7	拡張リンク要求		XFlag = H
0x40	01000000	b6	EPP モード要求		XFlag = H
0x30	00110000	b5 (+b4)	RLE 付き ECP モード要求		XFlag = H
0x10	00010000	b4	ECP モード要求		XFlag = H
0x08	00001000	b3	予約		XFlag = H
0x34	00110100	b2 (+b5+b4)	デバイス ID の返送要求	RLE 付き ECP モードで返送	XFlag = H
0x14	00010100	b2 (+b4)		ECP モードで返送	XFlag = H
0x05	00000101	b2 (+b0)		バイト・モードで返送	XFlag = H
0x04	00000100	b2		ニブル・モードで返送	XFlag = H
0x02	00000010	b1	予約		XFlag = H
0x01	00000001	b0	バイト・モード要求		XFlag = H
0x00	00000000	なし	ニブル・モード要求		XFlag = L

(注) ペリフェラル側のデコードの負担を軽くするため，1 ビットに対して 1 個の機能が割り当てられている．

〈表 2-7〉 拡張要求コード

可変長の ASCII 文字の列．列の長さ (バイト数) を示す 2 バイト整数 (MSB ファースト) が先頭に付加される．
ASCII 文字の列は，
キー：値；キー：値；…キー:値：
のように，キー名とそれに対応する値の組み合わせを並べたもの．
1 個のキーに複数の値が対応する場合，複数の値をカンマで区切って並べることができる．

キー名	省略形	値
MANUFACTURER	MFG	メーカ名を示す ASCII 文字列 (任意長)
COMMAND SET	CMD	対応言語を示す ASCII 文字列 (任意長)
MODEL	MDL	機種名を示す ASCII 文字列 (任意長)

(a) 必須のキー

"0x00,0x2E,MFG:ABC Corp.CMD:PCL,MPL;MDL:LaserPrinterDE-1:"

メーカ名 = ABC Corp.
"対応言語 = PCL,MPL"
機種名 = LaserPrinterDE-1

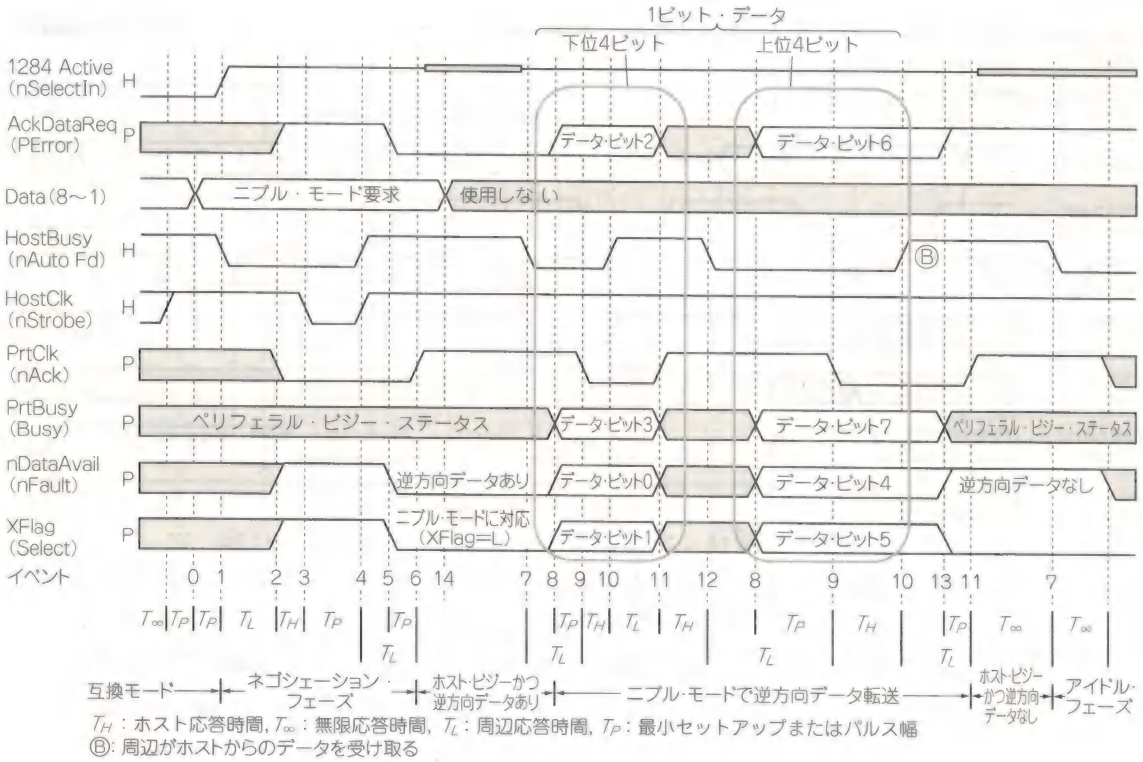
(b) デバイス ID の例

をHに，Host Busy (nAutoFd) をLにします (イベント 1)．それに対して，ペリフェラルが IEEE 1284 をサポートしていれば，AckDataReq (PError)，nDataAvail (nFault)，XFlag (Select) をともにHにして応答します (イベント 2)．**ペリフェラルがこれと異なる応答を返した場合は**，IEEE 1284 未対応とみなしてネゴシエーションを中止します．

ペリフェラルがこれと異なる応答を返した場合は
その中には，ペリフェラルが何も応答しなかった場合も含まれる．

ペリフェラルが IEEE 1284 対応であれば，ホストはネゴシエーションを開始します．データ線に 1 バイトの拡張要求コード (**表 2-6**) を出力し，HostClk (nStrobe) パルスを使ってペリフェラルに転送します (イベント 3～4)．拡張要求コードには，モード要求 (ニブル・モード要求，バイト・モード要求，ECP モード要求，RLE 付き ECP モード要求，EPP モード要求)，デバイス ID の返送要求 (ニブル・モードで返送，バイト・モードで返送，ECP モードで返送，RLE 付き ECP モードで返送)，拡張能力リンク要求などの種類があります．

〈図 2-9〉**IEEE1284 の電気的仕様**（H はホストが出力，P はペリフェラルが出力）



拡張要求コードを受け取ったデバイスは，**その機能に対応しているかどうかを** XFlag (Select) で応答します（イベント 5）．ニブル・モード要求だけは，ペリフェラルが XFlag＝L を返したときが肯定応答です．それ以外のすべての要求は，ペリフェラルが XFlag＝H を返したときが肯定応答です．肯定応答を返した場合は，さらに要求の種類に応じて次のように応答します．

モード要求の場合は，そのまま要求されたモードに移行します．

デバイス ID の返送要求の場合は，要求されたモードでデバイス ID を返送します．

デバイス ID はプリンタのメーカー，機種，対応するプリンタ言語などをホスト（パソコン）に知らせるためのものです（**表 2-7**）．

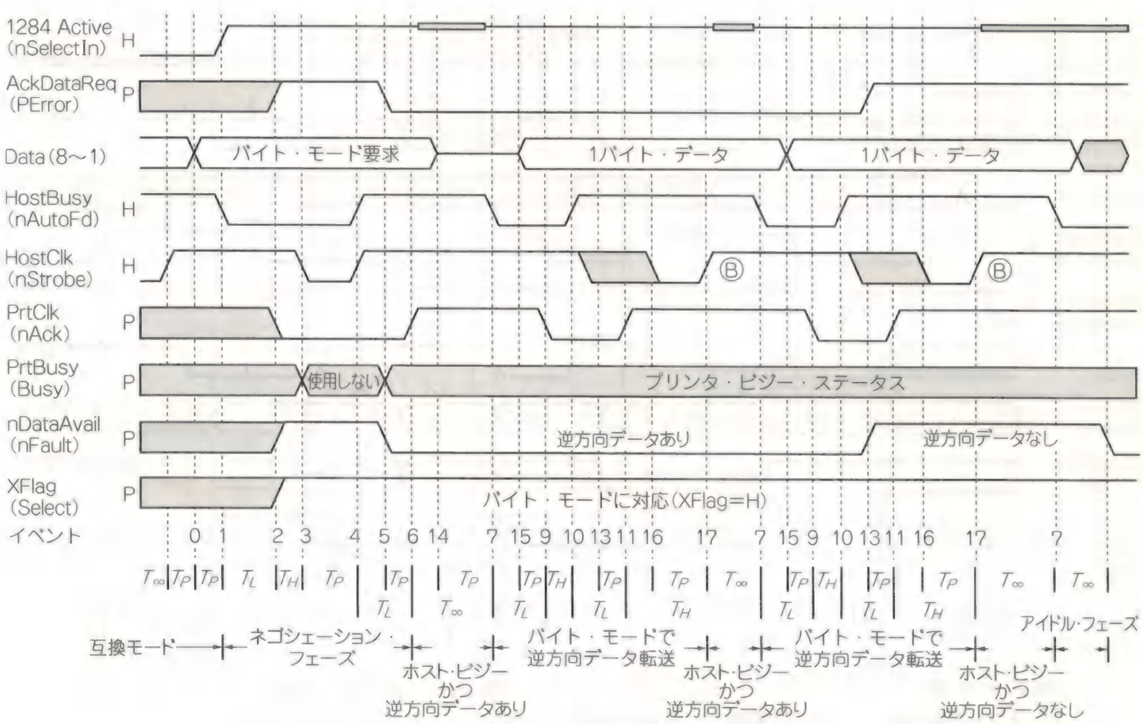
拡張能力リンクは，ペリフェラルがどの機能に対応しているかの問い合わせだけを行うもので，モード変更は行いません．ホストは問い合わせたいモードの拡張要求コードを次々に送ることができ，ペリフェラルは XFlag で応答します．

その機能に対応しているかどうかを
拡張要求コードは一度に一つの機能しか問い合わせできない．ホストやペリフェラルが複数の機能を備えている場合，最適のモードを発見するまで問い合わせと応答を繰り返す必要がある．これは，モード要求，デバイス ID 返送要求，拡張能力リンク要求のどれでも同じ．したがって，ホストは一度ネゴシエーションを実行してペリフェラルの応答をもらったら，それを記憶しておくべきだろう．同じペリフェラルに対して伝送のたびごとに問い合わせを繰り返すのはむだなことだ．

ニブル・モードのデータ転送

ニブル・モードでは，PrtBusy (Busy)，AckDataReq (PError)，XFlag (Select)，nDataAvail (nFault) の 4 本のステータス線を使って 4 ビットずつ逆方向のデータ転送を行います．したがって，2 回の転送で 1 バイトのデータを送

〈図 2-10〉 **バイト・モードで 2 バイトのデータ転送** (H はホストが出力，P はペリフェラルが出力)



ることができます (**図 2-9**).

ネゴシエーションでホストがニブル・モード要求 (コード＝00000000) を出し，ペリフェラルが XFlag＝L で肯定応答すると，自動的にニブル・モードに移行します (イベント 6). 同時に，ペリフェラルは nDataAvail (nFault) を用いて，出力したいデータがあるかどうかを示します.

nDataAvail＝ L (データあり) の場合，ホストは HostBusy＝L によってペリフェラルにデータ転送を要求します(イベント 7). ペリフェラルは PrtBusy(Busy)，AckDataReq (PError)，XFlag (Select)，nDataAvail (nFault) にデータの下位 4 ビット (D3 ～ D0) を出力し，PtrClk (nAck) パルスを使ってホストに転送します (イベント 9 ～ 11).

ホストは 4 ビットのデータを読み込むといったん HostBusy を H にしておき，HostBusy＝L によってペリフェラルに次のデータ転送を要求します (イベント 12). ペリフェラルは PrtBusy (Busy)，AckDataReq (PError)，XFlag (Select)，nDataAvail (nFault) にデータの上位 4 ビット (D7 ～ D4) を出力し，PtrClk (nAck) パルスを使ってホストに転送します (イベント 9 ～ 11). ホストは 4 ビットのデータを読み込むと HostBusy を H にします.

このように，ニブル・モードでは PrtClk (nAck) と HostBusy を使った 2 線ハンドシェイク方式でペリフェラル→ホストのデータ転送を行います.

ペリフェラルが出力したいデータがない場合 (nDataAvail＝H)，ホストはニブル・モードのままで nDataAvail が H になるのを待つ (ニブル・モード・アイドル・フェーズ) か，ニブル・モードを終了して互換モードに戻るかのいずれかを選ぶことができます.

ニブル・モード

ニブルとは 4 ビット (バイトの半分) という意味. 1 バイトのデータを 4 ビットずつ 2 回に分けて転送するので，この呼び名がついた. セントロニクス仕様やその他のパソコンのプリンタ・インターフェースは，一般に単方向伝送しかできない. だが，プリンタのステータス線を利用して逆方向の伝送を実現する方法がいろいろ工夫されてきた. IEEE 1284 のニブル・モードはその一つの方式を規格化したものだ.

ペリフェラルが出力したいデータがない場合

ネゴシエーションはホスト側から一方的に開始される. モード要求を受けたペリフェラル側はとくに出力したいデータをもたない場合や，データの準備が間に合わない場合もある. また，逆にペリフェラル側に出力したいデータがあっても，ホスト側からネゴシエーションをかけてくれなければ，いつまでも待たされてしまう.

バイト・モードのデータ転送

バイト・モードでは，Data8～Data1の8本のデータ線を使って8ビットの逆方向データ転送を行います（**図2-10**）．**バイト・モード**に対応するためには，ホストのハードウェアがデータ出力をオフ（ハイインピーダンス）にできることが必要です．

ネゴシエーションでホストがバイト・モード要求（コード＝00000001）を出し，ペリフェラルがXFlag＝Hで肯定応答すると，自動的にバイト・モードに移行します（イベント6）．同時に，ペリフェラルはnDataAvail（nFault）を用いて，出力したいデータがあるかどうかを示します．

nDataAvail＝L（データあり）の場合，ホストはHostBusy＝Lによってペリフェラルにデータ転送を要求します（イベント7）．ペリフェラルはData8～Data1にデータを出力し，PtrClk（nAck）パルスを使ってホストに転送します（イベント9～11）．ホストは8ビットのデータを読み込むとHostBusyをHにします．nDataAvail＝L（データあり）の状態が続いていれば，HostBusy＝Lによってペリフェラルに次のバイトの転送を要求します（イベント7）．

このように，バイト・モードでもPrtClk（nAck）とHostBusyを使った2線ハンドシェイク方式でペリフェラル→ホストのデータ転送を行います．

ペリフェラルが出力したいデータがない場合（nDataAvail＝H），ホストはバイト・モードのままでnDataAvailがHになるのを待つ（バイト・モード・アイドル・フェーズ）か，バイト・モードを終了して互換モードに戻るかのいずれかを選びます．

バイト・モード
PS/2から採用された双方向パラレル・ポートによって実現が可能になった．パソコン側では8ビットのデータ出力をオフ（ハイ・インピーダンス）にして，そこからデータを読み込む能力が必要．同様に，プリンタ側にも8ビットのデータ入力を出力に切り替える能力が必要．

ECPモードのデータ転送

ECPモードでは，Data8～Data1の8本のデータ線を使って8ビットの双方向データ転送を行います（**図2-11**，p.156）．

ネゴシエーションでホストが**ECPモード要求（コード＝00010000）**を出し，ペリフェラルがXFlag＝Hで肯定応答すると，自動的にEPPモードに移行します（イベント6）．

ECPモードでは，nReverseRequestとnAckReverseを用いて転送の方向を制御します．ホストがnReverseRequest＝Hで順方向（ホスト→ペリフェラル）を要求し，ペリフェラルがnAckReverse＝Hで応答すれば，順方向のデータ転送が行われます．ホストがnReverseRequest＝Lで逆方向（ペリフェラル→ホスト）を要求し，ペリフェラルがnAckReverse＝Lで応答すれば，逆方向のデータ転送が行われます．

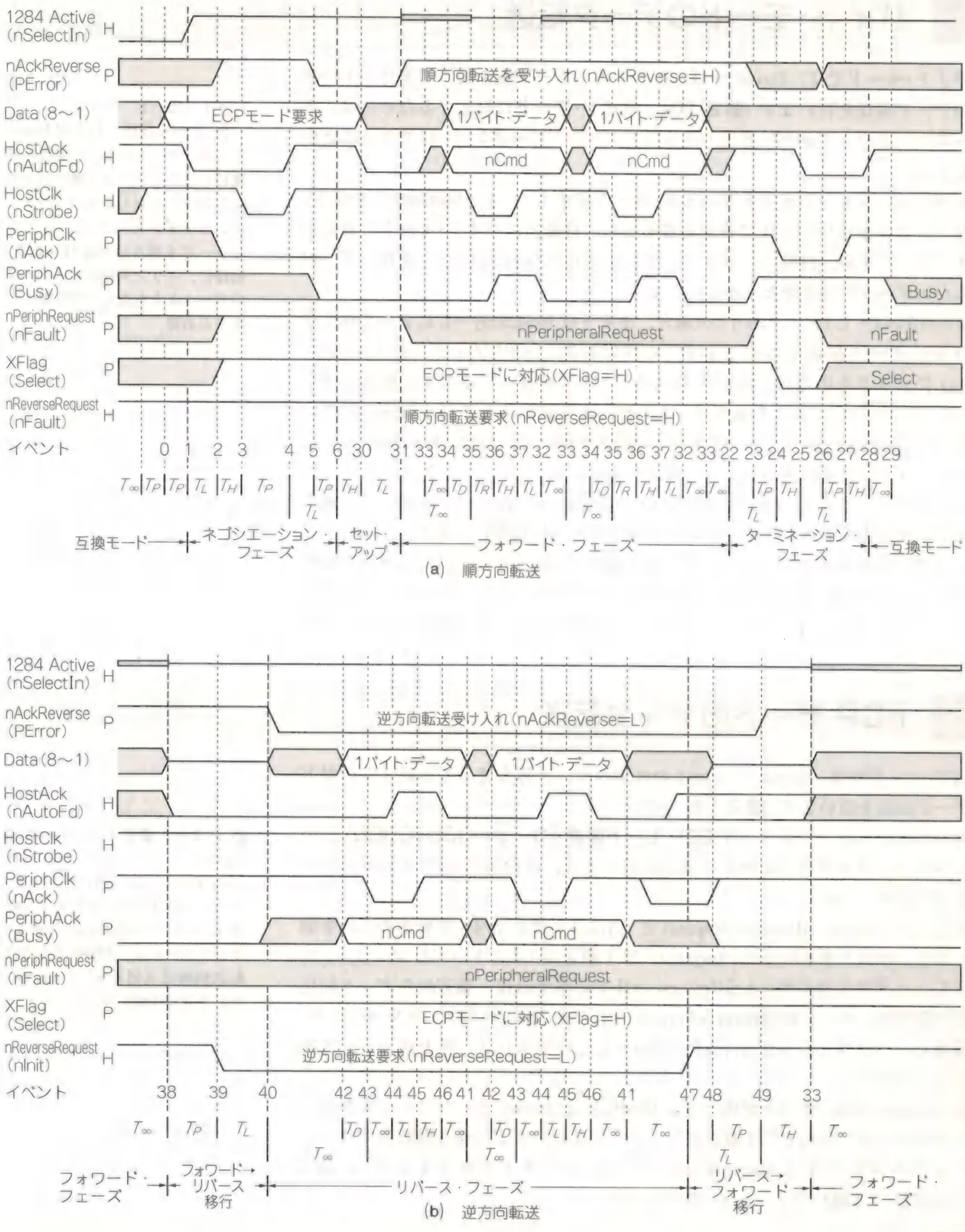
順方向転送では，ホストが出力するHostClk（nStrobe）とペリフェラルが出力するPeriphAck（Busy）で2線ハンドシェイクを行います．逆方向転送では，ペリフェラルが出力するPeriphClk（nAck）とホストが出力するHostAck（nAutoFd）で2線ハンドシェイクを行います．

ECPモード要求（コード＝00010000）
ECPモードに移行するには，もう一つRLE付きECPモード要求（コード＝00110000）もある．ネゴシエーション手順や基本的な転送制御は共通だが，データをRLE方式で圧縮して送る．

EPPモードのデータ転送

EPPモードの動作はほかのモードとはかなり異なり，一般的なCPUバスのデータ転送に近い方式となっています（**図2-12**）．すなわち，ホストがアドレス

〈図 2-11〉 ECP モードでデータ転送 (Hはホストが出力，Pはペリフェラルが出力)



〈表 2-8〉IEEE 1284 電気的仕様

項目	記事	min	max	条件
ドライバ特性	絶対定格	− 0.5V	+5.5V	出力開放時
	H 出力電圧	+2.4V	−	I_{oH} = -320 μ A
	L 出力電圧	−	+0.4V	I_{oL} = 14mA
	プルアップ抵抗（注）	1.8k Ω ± 5%	−	制御/ステータス線
		1k Ω ± 5%	−	データ線
レシーバ特性	絶対定格	− 2.0V	+7.0V	この範囲の過渡電圧には耐えられること
	H 入力電圧	−	+2.0V	
	L 入力電圧	+0.8V	−	
	H 入力電流	−	− 320 μ A	V_{iH} = 2.0V
	L 入力電流	−	12mA	V_{iL} = 0.8V
	プルアップ抵抗（注）	470 Ω ± 5%	−	制御/ステータス線
		1k Ω ± 5%	−	データ線
	入力容量	−	50pF	浮遊容量を含む
	入力立ち上がり時間	−	120ns	ケーブル長 2m

（注）プルアップ抵抗を使用する場合の値

(a)　レベル 1 デバイス
レベル 1 デバイスは TTL のように H，L のインピーダンスが非対称のもの．
ドライバ，レシーバとも 5 V TT レベルに従うものとする

項目	記事	min	max	条件
ドライバ特性	絶対定格	− 0.5V	+5.5V	出力開放時
	H 出力電圧	+2.4V	−	I_{oH} = − 14mA
	L 出力電圧	−	+0.4V	I_{oL} = 14mA
	出力インピーダンス	45 Ω	55 Ω	V_{oH} と V_{oL} の中点において
	スルーレート	50V/ μ s	400V/ μ s	
レシーバ特性	絶対定格	− 2.0V	+7.0V	この範囲の過渡電圧には耐えられること
	H 入力電圧	−	+2.0V	
	L 入力電圧	+0.8V	−	
	入力ヒステリシス電圧	0.2V	1.2V	
	H 入力電流	−	− 20 μ A	V_{iH} = 2.0V
	L 入力電流	−	20 μ A	V_{iL} = 0.8V
	入力容量	−	50pF	浮遊容量を含む

（注）プルアップ抵抗を使用する場合の値

(b)　レベル 2 デバイス
レベル 2 デバイスは CMOS のように H，L のインピーダンスが対称のもの．
ドライバ，レシーバとも 5VTTL レベルに従うものとする

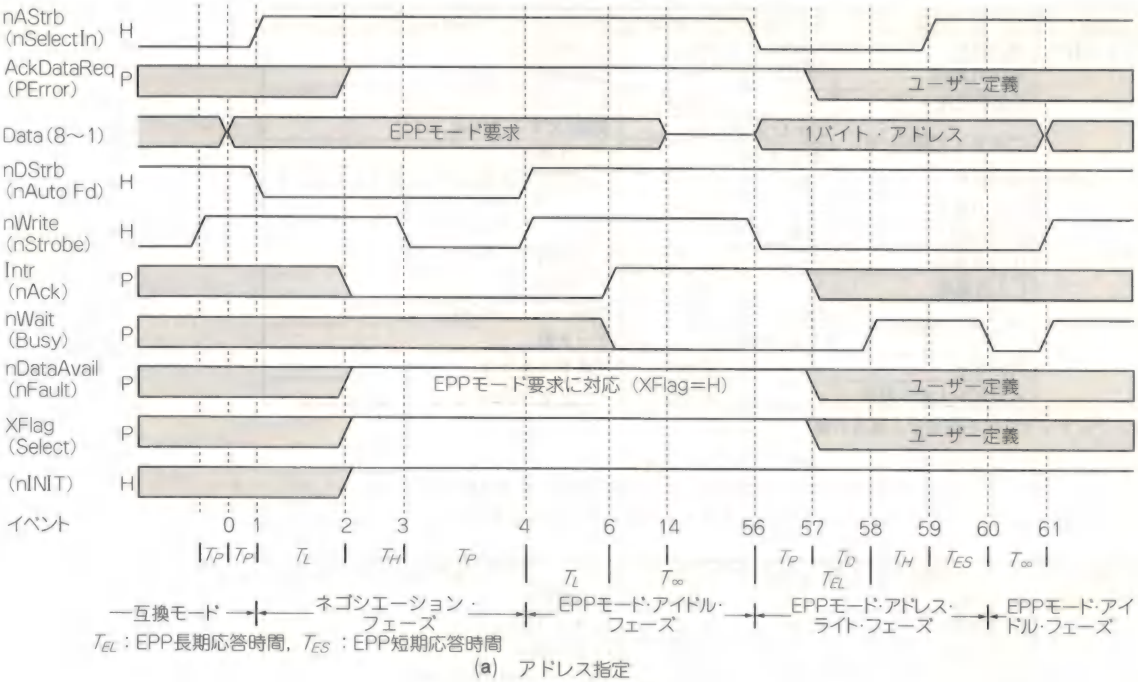
トランジスタ技術 エレクトロニクスの基礎と実用技術を凝縮したフィールド・ワーク・マガジン
SPECIAL No.70
B5 判　160 頁
定価 1,835 円（税込）

特集　IEEE1394 で広がる通信技術
パソコン周辺機器から情報家電のインターフェースまで

IEEE1394 は 1980 年代にアップル社の技術者が発想した FireWire というシリアル・バス規格がおおもとです．その後，AV 機器のディジタル・インターフェースとして iLink，電子楽器用のディジタル・インターフェースとして mLAN，PC への応用として OHCI，SBP2 などが提案され，製品に搭載されています．AV 機器間や AV 機器とパソコンを接続する通信インターフェースとして，IEEE1394 はもっとも有力なものであると思います．
本書では，ケーブルとコネクタといった基本的な部分から，IEEE1394 で LAN を構成するところまでを解説します．

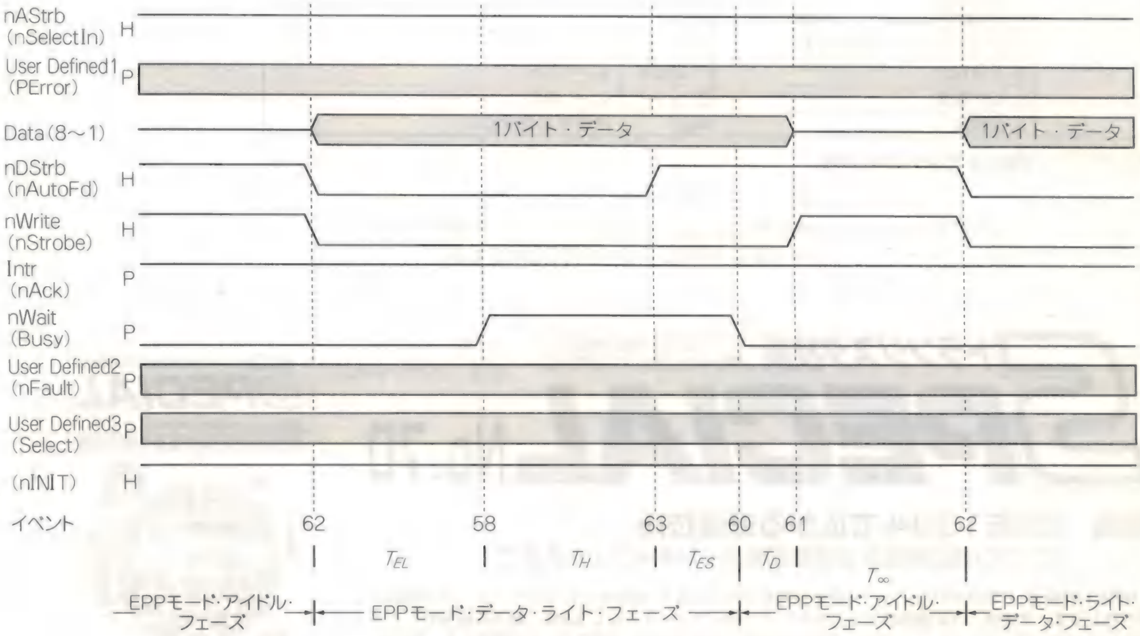
CQ出版社
〒 170-8461
東京都豊島区巣鴨 1-14-2　販売部　TEL（03）5395-2141　振替 00100-7-10665

〈図 2-12〉 **EPP モードでデータ転送** (H はホストが出力，P はペリフェラルが出力)

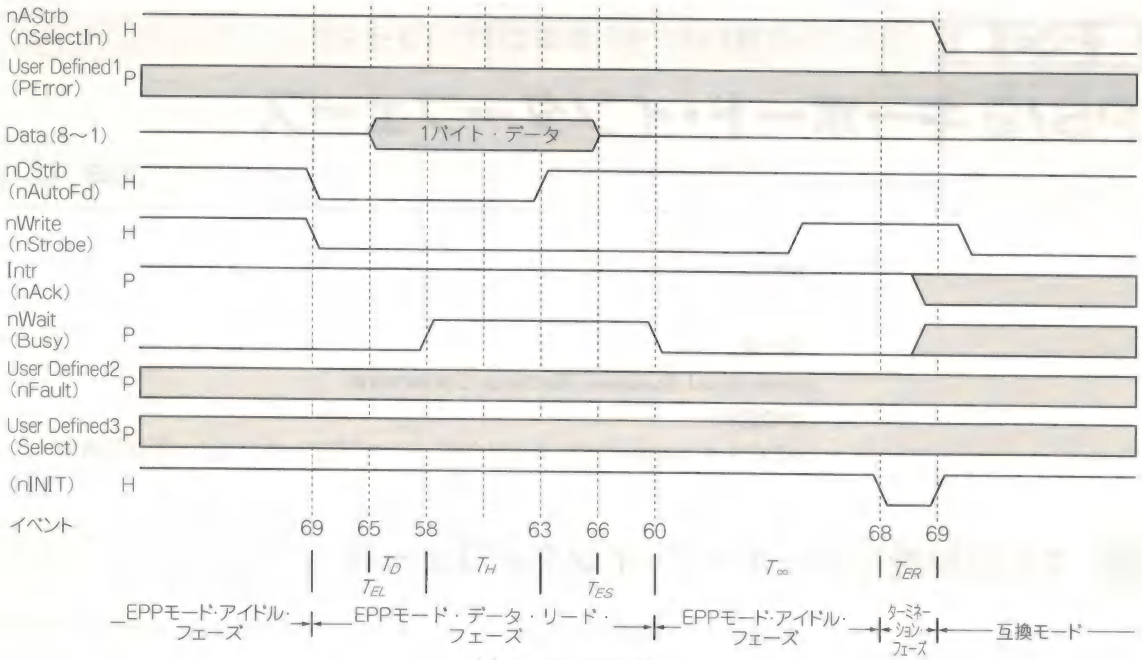


T_{EL}：EPP長期応答時間，T_{ES}：EPP短期応答時間

(a) アドレス指定



(b) データ・ライト



(c) データ・リード

を出してペリフェラルの内部レジスタを指定して，データをリード/ライトします．そのため，Data8 ～ Data1 の 8 本のデータ線を，8 ビット・アドレスと 8 ビット・データの両方に用います．

ネゴシエーションでホストが EPP モード要求（コード＝01000000）を出し，ペリフェラルが XFlag＝H で肯定応答すると，自動的に EPP モードに移行します（イベント 6）．

EPP モードでは，データの転送方向とは関係なく，**ホストが出力するストローブ信号に同期して**アドレスやデータが転送されます．アドレス転送の場合は nAStrb（nSeletcIn），データ転送の場合は nDStrb（nAutoFd）をホストが出力します．また，ホストが出力する nWrite（nStrobe）によって転送の方向（リード/ライト）が変わります．さらに，ペリフェラル側からは nWait（Busy）を用いて転送のサイクル時間を延長できます．

ホストが出力するストローブ信号に同期して
他のモードはすべて非同期のハンドシェイク転送を行うが，この EPP モードは同期転送に分類することができる．

電気的仕様

電気的仕様は，レベル 1 デバイスとレベル 2 デバイスの二つに分けられます（表 2-8）．

レベル 1 デバイスは，通常の TTL デバイスのように H，L のインピーダンスが非対称のもので，セントロニクス仕様や PC/AT 仕様に対してほぼ上位互換となっています．

レベル 2 デバイスは，CMOS デバイスのように H，L のインピーダンスが対称で，かつレシーバが高入力インピーダンスとなっているものです．この場合も，ロジック・レベルは TTL レベルに従います．ただし，ECP や EPP などの高速伝送時に反射による波形歪みを生じるのを防ぐため，インピーダンス整合やドライバの出力スルーレートの制限が規格に加えられています．

第2-3章 ユーザから見れば，そう簡単にはなくならない

PS/2キーボード・インターフェース

宮崎　仁

〈発行日〉
1987年
〈発行者〉
International Business Machines Corporation
〈参考資料〉
OADGテクニカル・リファレンス（PCオープン・アーキテクチャ推進協議会）

AT互換機のキーボード・インターフェース

PS/2
1987年にIBM社が発売したPC/ATの後継機．PC/ATとは基本アーキテクチャを若干変更し，かつ情報の公開をやめて互換機を作れなくした．しかし，PS/2で採用された技術は現在のAT互換機にもかなり受け継がれている．

DINコネクタ
小型，多ピンの丸形コネクタ．DINはDeutsches Institut fuer Normungの略．ドイツを代表する規格で，日本のJIS，アメリカのANSIに相当する．

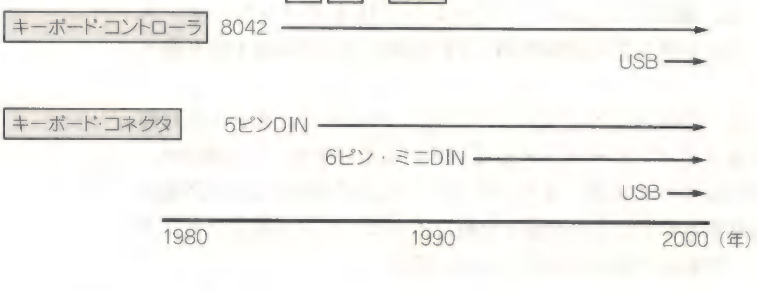
現在のAT互換機のキーボード・インターフェースとしては，**PS/2**から採用されたものが一般的に採用されています．しかし，インターフェースの基本的な方式は，オリジナルのPC/XTやPC/ATからほとんど変わっていません（**図3-1**）．

PC/ATのキーボード・インターフェースは，本体側にキーボード・コントローラとして8ビット・シングルチップ・マイコンの8042を内蔵し，キーボード・ユニット側にも同クラスの8ビット・シングルチップ・マイコン（たとえば8048，8051，6805など）を内蔵するインテリジェントなものです．そして，本体とキーボードの間はデータ線（DATA），クロック線（CLK），5V，GNDのシンプルな4線インターフェースで接続されています．

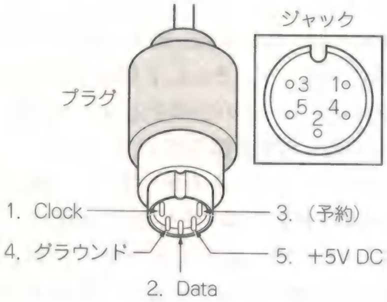
1984年に発売されたPC/ATでは5ピン**DINコネクタ**（**図3-2**）を採用していました．それが，1987年に発売されたPS/2では6ピン・ミニDINコネクタ（**図3-3**）に変わりました．一般に，前者はATキーボード，後者はPS/2キーボードと呼ばれます．どちらも実際に使われているピンは4本だけで，余りは予約となっています．両者の違いはコネクタだけなので，DIN－ミニDIN変換アダプタを使えば相互に接続できます．

このようにコネクタが変更になった一つの理由は，PS/2ではキーボード・コントローラの8042を拡張してマウス・インターフェースを標準装備したことです．

〈図3-1〉AT互換機のキーボード・インターフェース



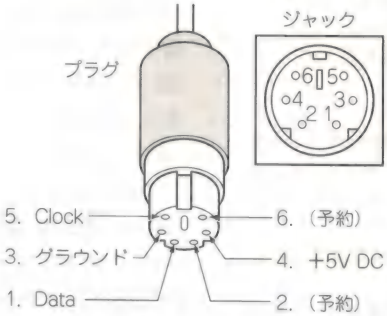
〈図3-2〉[1] ATタイプのキーボード・コネクタ（5ピンDIN）



コネクタのピン割り当て

ピン	I/O	信号名
1	I/O	Clock(CLK)
2	I/O	Data(DATA)
3	—	(Reserved)
4	—	グラウンド
5	—	+5V DC

〈図3-3〉[1] PS/2タイプのキーボード・コネクタ（6ピン・ミニDIN）



コネクタのピン割り当て

ピン	I/O	信号名
1	I/O	Data(DATA)
2	—	(Reserved)
3	—	グラウンド
4	—	+5V DC
5	I/O	Clock(CLK)
6	—	(Reserved)

PC/ATはマウス・インターフェースを装備しておらず，**シリアル・マウス**が一般的に使われていました．このPS/2マウスのコネクタに6ピン・ミニDINが採用されたため，PS/2キーボードのコネクタもそれに合わせて変更されました．

AT互換機では，当初はATキーボード・インターフェースを採用するものが多かったのですが，最近はより小型のPS/2キーボード・インターフェースが一般的になりました．PC97ハードウェア・デザインガイドでは，キーボードとマウスはUSB接続を推奨していますが，そうでなければPS/2仕様が使われます．

実際のパソコン製品には，さまざまなものが混在しています．デスクトップのマザーボードでは，USBポート，PS/2キーボード，PS/2マウスをすべて備えるものがもっとも多いようですが，PS/2キーボードとPS/2マウスを省略して**USBだけにしたものもあります**．ノートブックはもともとキーボードは内蔵しており，マウスに代わるポインティング・デバイスも内蔵しています．拡張性を重視した比較的大型のノートブックでは，やはりUSBポート，PS/2キーボード，PS/2マウスをすべて備えるものもあります．省スペースを重視して，PS/2キーボードとPS/2マウスを省略してUSBだけにする場合も多いようです．

単売のキーボード，マウスでも，最近はPS/2用とUSB用がほぼ半々ぐらいの割合で見られます．現在はPS/2からUSBに移行する過渡期ですが，キーボードやマウスをUSB接続にしてみてもユーザはとくに便利になるわけではないので，PS/2キーボードやPS/2マウスはそう簡単にはなくならないでしょう．

シリアル・マウス
専用ポートではなく，シリアル・ポートに接続するタイプのマウス．

USBだけにしたものもあります
ISAおよびシリアル・ポート，パラレル・ポート，PS/2キーボード，PS/2マウスはレガシー・インターフェースと呼ばれ，これらをもたないAT互換機はレガシー・フリーPCと呼ばれます．シリアル・ポート，パラレル・ポート，PS/2キーボード，PS/2マウスはすべてUSBで代用できます．

AT互換機のキーボードの変遷

一方，キーボードのほうは次々にキーが増設され，変化してきました（**図3-4**）．

初代IBM PCやPC/XTは84キーボードを利用していましたが，PC/ATで**キーの数を増やした**101キーボードが追加され，PS/2やその後のAT互換機でも引き続き標準として使われてきました．最近では，Windowsの普及に合わせて，2個の**windowsキー**と1個の**applicationキー**を追加した104キーボード

キーの数を増やした
84キーに対して，101キーではファンクション・キーを増設（10個→12個）し，カーソル・キーやINS，DEL，HOME，END，PgUp，PgDnなどのキーをテン・キーから独立させた．

が標準となっています．これらは，いずれも英語版のキーボードです．

一方，日本語版のキーボードにはかな漢字変換用のキーが必要になるため，キーの数が増えます．日本語版 AT 互換機（いわゆる DOS/V パソコン）の仕様を標準化する団体である OADG（Open Architecture Development Group）では，101 キーボードをベースに日本語化した 106 キーボード，それに 2 個の windows キーと 1 個の application キーを追加した 109 キーボードの仕様を定めています．106 キーボードのキー配置，刻印とキー番号の割り当てを**図 3-5** に示します．

なお，英語 OS の DOS は当然英語版の 101 キーボードをサポートし，日本語 OS の **DOS/V は当然日本語版の 106 キーボードを**サポートします．106 キーボードは 101 キーボードをベースにしていますが，記号キーなどの位置は **PC9800 シリーズからの移行を意識して**かなり変更されています．DOS を英語モードに切り替えたり，何かのトラブルでモードが切り替わってしまったときなど，キーボードの設定が変わってキーの刻印と違う記号が出てくることがあるので注意が必要です．

さて，インターネットの急速な普及とともに，**インターネット端末として利用するためだけの目的で**パソコンを購入するユーザが多くなってきました．また，CD プレーヤ，DVD プレーヤ，MP3 プレーヤなどのマルチメディア機器としてパソコンを利用するユーザも増えています．

そのようなユーザが簡単にパソコンを利用できるように，大手メーカのホームユース向けパソコンを中心に，Web ブラウザを起動してインターネットに接続するための専用ボタンや，動画，音声の再生，停止や音量調整などの専用ボタンを備えたキーボードが増えてきました．また，それ以外にも特定アプリケーションを一発で起動するための専用ボタンや，ユーザが任意のアプリケーションに割り当てられるプログラマブルなアプリケーション・ボタンを備えたものもあります．

とくに，Microsoft 社ではこのところソフトウェア製品だけでなくマウスやキーボードなどの入力用機器を積極的に販売しています．最初の頃は，従来製品と機能的には同一で，形状などに人間工学を取り入れたマウスやキーボードを製品化していました．最近は，**ホイール**や専用ボタン（**ホット・キー**）などの機能を取り入れたマウスやキーボードの製品化や普及を積極的に進めています．キーボードでは，インターネット用やマルチメディア用の専用ボタンを備えたマイクロソフト・インターネット・キーボードがあります．リプレース市場でユーザに直接販売するだけでなく，大手パソコン・メーカにも OEM 供給しており，Microsoft 社の拡張キーボードが次第にデファクト・スタンダードになっていくことも考えられます．

windows キー
押すと，Windows のスタート・メニューが出てくる．また，他のキーと組み合わせてショート・カットの機能をもつ．

application キー
押すと，現在選択されているアイコンのメニューが出てくる．マウスの右ボタンに相当する．

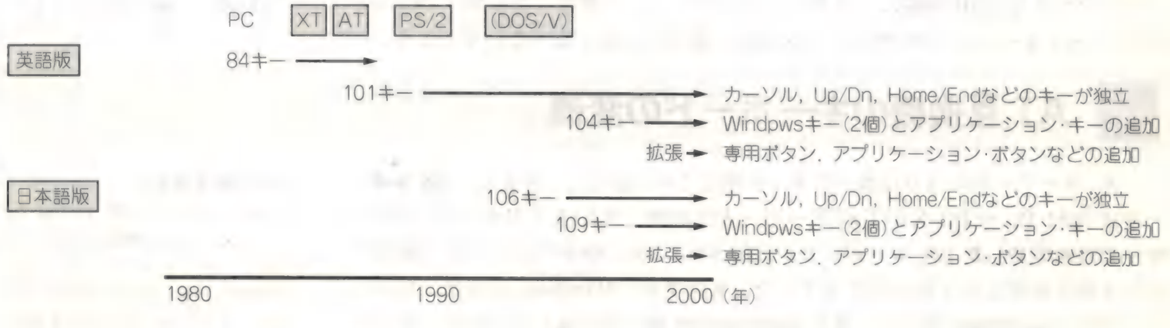
DOS/V は当然日本語版の 106 キーボードを
DOS/V には日本語モードと英語モードがあり，モードを切り替えるとサポートするキーボードも変わる．

PC9800 シリーズからの移行を意識して
もともと，日本 IBM では PC/AT ベースではなく，PS/2 をベースにして日本語化した PS55 シリーズを販売していた．このとき，PC9800 シリーズや JIS キーボードを強く意識した独自のキーボードを採用した．106 キーボードは，この独自キーボードの影響を強く受けている．

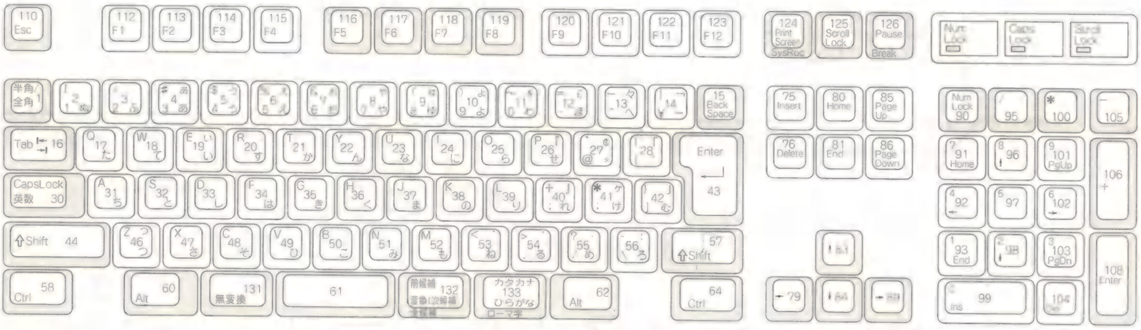
インターネット端末として利用するためだけの目的で
パソコンをコンピュータではなく家電として利用するようになってきた，ということ．

ホイール
画面スクロールが簡単にできるというホイール付きマウスは，あっという間に市場にあふれてきた．使わなくてもじゃまにはならないが，コストはだいぶ高そうだ．

〈図 3-4〉 AT 互換機のキーボード



〈図 3-5〉106 キーボードの配列



ただし，このような機能拡張は，パソコン本体側ではすべてドライバの追加によってソフトウェア的に処理できます．キーボード・インターフェース自体には何も変更がありません．

ホット・キー
Microsoft 社では，この「便利な」専用ボタンをホット・キーと呼んでいる．キーが熱いと何かいいことがあるのだろうか．

キーボード・インターフェースの仕様

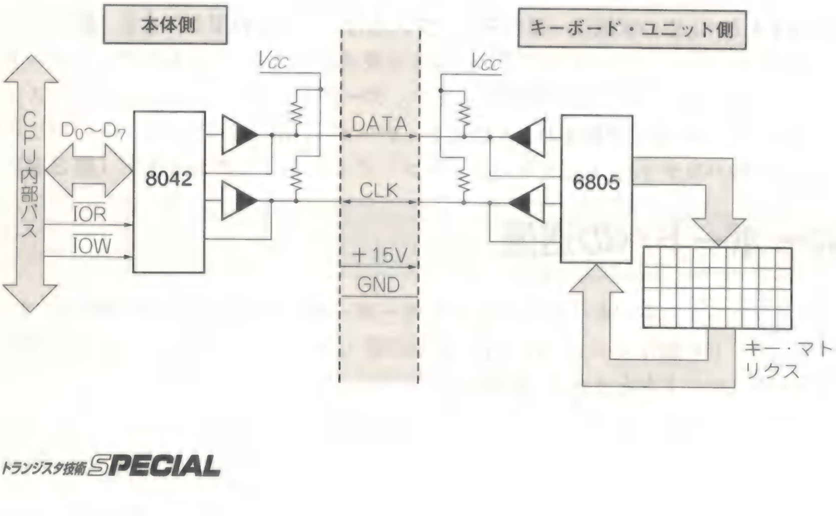
PC/AT のキーボード・インターフェースは，データ線（DATA），クロック線（CLK），5 V，GND の 4 本を使うシンプルな**クロック同期式のシリアル・インターフェース**によって実現されています（**図 3-6**）．5 V と GND はキーボードへの電源供給に使うことができます．PS/2 では同じ 8042 にマウス・インターフェースの機能が追加されましたが，キーボード・インターフェースの部分はほぼ同じです．

クロック同期式のシリアル・インターフェース
クロック線のパルスに同期してデータ線の信号を変化させる方式．

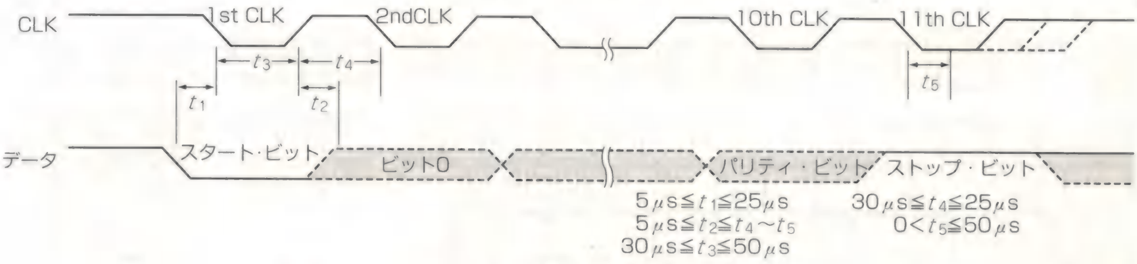
DATA も CLK もパソコン側とキーボード側が対称な双方向回路になっています．両出力ともオープン・コレクタのバッファ（オリジナルの PC/AT では 7407 相当）が入っており，双方が同時に信号を出力しても壊れることはありません．信号が出力されていないときは，プルアップ抵抗によって信号線は H に保たれます．信号レベルは L が 0.7 V 以下，H が 2.4 V 以上と規定されています．

このシリアル・インターフェースは 8042 のビット入出力を利用してソフトウェ

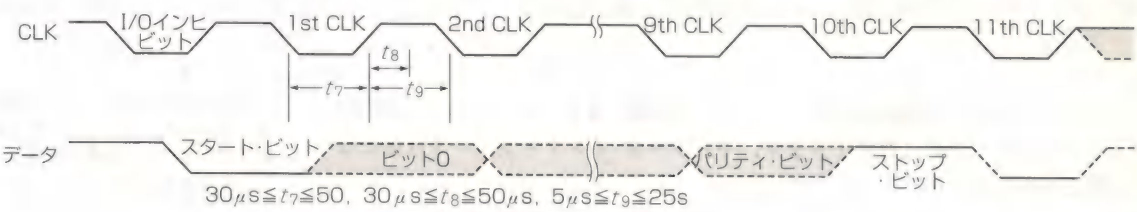
〈図 3-6〉PC/AT キーボード・インターフェース



〈図 3-7〉送受信データのタイミング



(a) 受信データのタイミング



(b) 送信データのタイミング

〈表 3-1〉[1] CLK/DATA 信号線の状態と意味

CLK	DATA	STATUS	内　容
"L"	——	キーボード禁止	送信データをキーボード内のバッファにためておく
"H"	"L"	システム送信要求	送信データをキーボード内のバッファにためておく システムからデータを受信する
"H"	"H"		データを送信する
"L"	——	回線競合	システムがCLKを"L"にすると通信を中断できる

ア的に実現されており，DATAやCLKのタイミングにはかなり幅をもたせてあります（**図 3-7**）.

パソコンはDATA線を使ってキーボードにコマンドを書き込んだり，キーボードのデータを読み出したりできます．パソコン→キーボードの転送時は，キーボード側で**DATA出力をオフ(H)**にします．キーボード→パソコンの転送時は，パソコン側でDATA出力をオフ(H)にします．いずれの場合も，CLKはキーボード側が出力し，パソコン側はCLK出力をオフ(H)にします．ただし，パソコン側がCLK出力を強制的にLにすることによって，キーボードの出力を禁止したり，そのとき行われている転送を中止することができます．このように，DATAとCLKの状態は一種の**ステータス信号**としても利用されます（**表 3-1**）.

DATA出力をオフ(H)に
オープン・コレクタ出力なので，トランジスタをオフにすればプルアップ抵抗によって電圧はハイになり，相手側の出力で信号電圧を決められるようになる．

ステータス信号
DATAとCLKの組み合わせによって，システムの現在の状態（ステータス）を示すことができる．

パソコンとキーボードの間では，調歩同期式に似た11ビットのデータ・ストリームを使ってデータをやり取りします．データ・ストリームは1ビットのスタート・ビット(L)で始まり，8ビット・データ（LSB先行），1ビットのパリティ（奇数パリティ），1ビットのストップ・ビット(H)で終わります（**図 3-8**）.

パソコンからキーボードへの送信

パソコンからキーボードへの送信は，キーボードに対するコマンドの転送です．キーボードの動作モードの設定や，初期状態へのリセット，データの再送信要求などのコマンドがあります（**表 3-2**）.

〈図 3-8〉シリアル・データのフォーマット

スタート・ビット	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	パリティ・ビット	ストップ・ビット

- スタート・ビット ―― 1ビット
- データ長 ―― 8ビット
- パリティ・ビット ―― ODDパリティ
- ストップ・ビット ―― 1ビット

パソコンは，別のデータの転送中を除いて，いつでもキーボードに対してコマンドを送信できます．また，キーボードから送信中であっても，10 クロック目に入る前であれば CLK を L にして転送を中止できます．**10 クロック目に入ったら**，その転送は最後まで完了しなければならないと規定されています．

ただし，パソコンが送信を行うためにはキーボード側から CLK を出力してもらわなければならないので，実際には送信要求のステータス（DATA＝L，CLK＝H）を出してキーボードの応答を待ちます．

キーボードは，10 ms 以下の間隔でステータスを調べ，送信要求を検出したらそれに応答しなければなりません．

10 クロック目に入ったら
転送は 11 クロックで完了するので，ここまできたら中止するより最後まで実行してしまったほうがよい．

トランジスタ技術 SPECIAL エレクトロニクスの基礎と実用技術を濃縮したフィールド・ワーク・マガジン
B5 判　172 頁
定価 1,840 円（税込）
CQ出版社

SPECIAL No.69

特集 作ってわかる通信ネットワーク技術
携帯電話や自作 LAN を利用して情報収集をしよう

21 世紀は通信とネットワークの時代だと言われています．現今の消費不況の中でも，携帯電話を初めとした情報通信費は伸び続けていると聞きます．今回の特集では，商用通信回線の携帯電話やパケット通信端末，計測・制御などへの利用を目的とした 1 線式バス・システム MicroLAN などを使って，データ収集の新しい方策を探ります．また，TCP/IP プロトコルをハードウェアで内蔵した LSI を使って，簡単な通信実験システムを作ります．

第 1 章　インターネットを代表とする通信の世界を見る
21 世紀は通信とネットワークの時代だ

第 2 章　配線も工事もいらない…
携帯電話リモート計測制御システムの製作

第 3 章　温度，湿度，気圧，日射量，紫外線量をリモート観測する
携帯電話による無人気象観測システムの製作

Appendix　Davis 社 Weather Station による気象観測

第 4 章　1 線式バス・システムで Windows 計測システムを作る
温度などの計測や水量などの制御が簡単にできる

第 5 章　1 線式バス・システムのマイコン組み込みシステムへの応用
パソコンを開発システムに使って計測システムを作る

第 6 章　組み込みシステム用 TCP/IP 接続環境の製作
ネットワーク・プロトコル・スタック IC S-7600A を使った

CQ出版社　〒170-8461　東京都豊島区巣鴨 1-14-2 販売部　☎ (03) 5395-2141　振替 00100-7-10665

〈表 3-2〉[1]コマンド一覧（システム→キーボード）

コード	コマンド名	内　容	応　答
F5h	デフォルト・ディセーブル	電源投入時のデフォルト状態にリセット 走査停止し，次のコマンドを待つ	ACK
EEh	エコー	診断用	(EEh)
F4h	イネーブル	出力バッファとタイプマティックを停止し走査開始	ACK
F2h	ID 読み出し	識別情報を要求	ACK (ID- "L") (ID- "H")
FEh	再送信	キーボードに再送信要求	(直前のバイト)
FFh	リセット	リセットと自己診断	ACK
F0h	代替走査コード選択	走査コードの選択　[F0h] [ACK] [Opt.] [ACK] Opt.=00 ScanCode get 現在の ScanCode を返す =01 ScanCode set 1 =02 ScanCode set 2 =03 ScanCode set 3	 ACK (Code) ACK × ACK ACK × ACK ACK × ACK
F7h	セット・オール・キー	すべてのキーをタイプマティックにセットする	ACK
F8h	セット・オール・キー	すべてのキーをメイク/ブレークにセットする	ACK
F9h	セット・オール・キー	すべてのキーをメイクにセットする	ACK
FAh	セット・オール・キー	すべてのキーをタイプマティック/メイク/ブレークにセットする	ACK
F6h	セット・デフォルト	電源投入時のデフォルト状態に戻す．走査は継続される	ACK
FBh	セット・キー・タイプ	個別のキーをタイプマティックにセットする [FBh] [ACK] [Scan Code] … [Scan Code]	ACK
FCh	セット・キー・タイプ	個別のキーをメイク/ブレークにセットする [FCh] [ACK] [Scan Code] … [Scan Code]	ACK
FDh	セット・キー・タイプ	個別のキーをメイクにセットする [FDh] [ACK] [Scan Code] … [Scan Code]	ACK
FDh	セット/リセット・ステータス・インジケータ	キーボード上の NumLock CapsLock ScrolllLock LED の ON/OFF　[EDh] [ACK] [Opt.] [ACK] Opt.=Bit0 ： Scroll Lock　1=ON/0=OFF Bit1 ： Num Lock Bit2 ： Caps Lock Bit3 ： Kana Indicator (AX-Keyboard)	ACK ACK × ACK
F3h	セット・タイプマティック・レート/ディレイ	タイプマティックのレートとディレイ設定 [E3h] [ACK] [Opt.] [ACK] Opt.=Bit0 ～ 4 ：レート（別表 B） =Bit5 ～ 6 ：ディレイ（別表 A）	ACK × ACK

〈別表 A〉

ビット 56	時間
00	250ms
01	500ms
10	750ms
11	1000ms

〈別表 B〉

ビット 4～0	タイプマティック・レート	ビット 4～0	タイプマティック・レート	ビット 4～0	タイプマティック・レート	ビット 4～0	タイプマティック・レート
00000	30.0	01000	15.0	10000	7.5	11000	3.7
00001	26.7	01001	13.3	10001	6.7	11001	3.3
00010	24.0	01010	12.0	10000	6.0	11010	3.0
00011	21.8	01011	10.9	10011	5.5	11011	2.7
00100	20.0	01100	10.0	10100	5.0	11100	2.5
00101	18.5	01101	9.2	10101	4.6	11101	2.3
00110	17.1	01110	8.6	10110	4.3	11110	2.1
00111	16.0	01111	8.0	10111	4.0	11111	2.0

〈表 3-3〉[1]コマンド一覧（キーボード→システム）

コード	コマンド名	内　容
FAh	ACK 応答	エコーと再送信コマンド以外の有効な入力で返す
AAh	BAT 完了コード	BAT が正常に終了すると返す
FCh	BAT 障害コード	BAT 障害が生ずると返し，走査停止する
F0h	ブレーク・コード・プリフィックス	ScanCode set 2 または 3 を使用中のブレーク・コードの第 1 バイト
EEh	エコー	エコー・コマンドに対して返す
83ABh	キーボード ID	ID 読み出しで返すコード　[ID "L" :ABh] [ID "H" : 83h] テン・キーのないキーボードでは 84ABh を返す
00h	オーバラン	キー・バッファ容量を越えたとき，バッファの最後に入れられる
FFh	オーバラン	キー・バッファ容量を越えたとき，バッファの最後に入れられる
FEh	再送信	無用な入力，不正なパリティをもつ入力があると発行する

〈表3-4〉[1] 106キーボード・スキャン・コード

各キーに個別の8ビットのメイク・スキャン・コードが割り当てられ，キーが押されたときにシステムに送られる．キーが放されたとき，ブレーク・コード・プリフィックス（F0h）が付加されてシステムに送られる．ブレーク．コードは2バイトで構成され，最初のバイトがブレーク・コード・プリフィックス（F0h）で，第2バイトがそのキーのメイク・スキャン・コードとなる．タイプマティック・スキャン・コードは，そのキーのメイク・スキャン・コード．

下記に，キーボードのシフト状態やシステムの状況に影響されない．各キーのスキャン・コードを示す．
下記のキーは，シフト・キー（Ctrl, Alt, Shift）やNum Lock（ONまたはOFF）の状況に応じて，異なる一連のスキャン・コードをシステムに送る．ベース・スキャン・コードは他のキーと同じなので，ベース・スキャン・コードに付加コード（E0h）を追記して個別化している．

キー番号	メイク・コード	ブレーク・コード	キー番号	メイク・コード	ブレーク・コード
1	0E	F0 0E	49	2A	F0 2A
2	16	F0 16	50	32	F0 32
3	1E	F0 1E	51	31	F0 31
4	26	F0 26	52	3A	F0 3A
5	25	F0 25	53	41	F0 41
6	2E	F0 2E	54	49	F0 49
7	36	F0 36	55	4A	F0 4A
8	3D	F0 3D	56	51	F0 51
9	3E	F0 3E	57	59	F0 59
10	46	F0 46	58	14	F0 14
11	45	F0 45	60	11	F0 11
12	4E	F0 4E	61	29	F0 29
13	55	F0 55	62	E0 11	E0 F0 11
14	6A	F0 6A	64	E0 14	E0 F0 14
15	66	F0 66	90	77	F0 77
16	0D	F0 0D	91	6C	F0 6C
17	15	F0 15	92	6B	F0 6B
18	1D	F0 1D	93	69	F0 69
19	24	F0 24	96	75	F0 75
20	2D	F0 2D	97	73	F0 73
21	2C	F0 2C	98	72	F0 72
22	35	F0 35	99	70	F0 70
23	3C	F0 3C	100	7C	F0 7C
24	43	F0 43	101	7D	F0 7D
25	44	F0 44	102	74	F0 74
26	4D	F0 4D	103	7A	F0 7A
27	54	F0 54	104	71	F0 71
28	5B	F0 5B	105	7B	F0 7B
29（予約）	5D	F0 5D	106	79	F0 79
30	58	F0 58	108	E0 5A	E0 F0 5A
31	1C	F0 1C	110	76	F0 76
32	1B	F0 1B	112	05	F0 05
33	23	F0 23	113	06	F0 06
34	2B	F0 2B	114	04	F0 04
35	34	F0 34	115	0C	F0 0C
36	33	F0 33	116	03	F0 03
37	3B	F0 3B	117	0B	F0 0B
38	42	F0 42	118	83	F0 83
39	4B	F0 4B	119	0A	F0 0A
40	4C	F0 4C	120	01	F0 01
41	52	F0 52	121	09	F0 09
42	5D	F0 5D	122	78	F0 78
43	5A	F0 5A	123	07	F0 07
44	12	F0 12	125	7E	F0 7E
45（予約）	61	F0 61	131	67	F0 67
46	1A	F0 1A	132	64	F0 64
47	22	F0 22	133	13	F0 13
48	21	F0 21			

キー番号	ベース，またはShift+Num Lock メイク/ブレーク	Shift時 メイク/ブレーク	Num Lock時 メイク/ブレーク
75	E0 70 /E0 F0 70	E0 F0 12 E0 70 /E0 F0 70 E0 12*	E0 12 E0 70 /E0 F0 70 E0 F0 12
76	E0 71 /E0 F0 71	E0 F0 12 E0 71 /E0 F0 71 E0 12*	E0 12 E0 71 /E0 F0 71 E0 F0 12
79	E0 6B /E0 F0 6B	E0 F0 12 E0 6B /E0 F0 6B E0 12*	E0 12 E0 6B /E0 F0 6B E0 F0 12
80	E0 6C /E0 F0 6C	E0 F0 12 E0 6C /E0 F0 6C E0 12*	E0 12 E0 6C /E0 F0 6C E0 F0 12
81	E0 69 /E0 F0 69	E0 F0 12 E0 69 /E0 F0 69 E0 12*	E0 12 E0 69 /E0 F0 69 E0 F0 12
83	E0 75 /E0 F0 75	E0 F0 12 E0 75 /E0 F0 75 E0 12*	E0 12 E0 75 /E0 F0 75 E0 F0 12
84	E0 72 /E0 F0 72	E0 F0 12 E0 72 /E0 F0 72 E0 12*	E0 12 E0 72 /E0 F0 72 E0 F0 12
85	E0 7D /E0 F0 7D	E0 F0 12 E0 7D /E0 F0 7D E0 12*	E0 12 E0 7D /E0 F0 7D E0 F0 12
86	E0 7A /E0 F0 7A	E0 F0 12 E0 7A /E0 F0 7A E0 12*	E0 12 E0 7A /E0 F0 7A E0 F0 12
89	E0 74 /E0 F0 74	E0 F0 12 E0 74 /E0 F0 74 E0 12*	E0 12 E0 74 /E0 F0 74 E0 F0 12
キー番号	メイク/ブレーク	Shift時メイク/ブレーク	
95	E0 4A/E0 F0 4A	E0 F0 12 E0 4A/E0 F0 4A/E0 12*	

＊：左のシフト・キーが押されると，他のスキャン・コードとともにF0 12/12のシフト・ブレークとメイク・コードが送られる．右のシフト・キーが押されると，F0 59/59が送られる．
両方のシフト・キーが押されると，両方のスキャン・コードが他のスキャン・コードとともに送られる．

キー番号	メイク/ブレーク	Ctrl時，Shift時 メイク/ブレーク	Alt時 メイク/ブレーク
124	E0 12 E0 7C /E0 F0 7C E0 F0 12	E0 7C/E0 F0 7C	84/F0 84

キー番号	メイク・コード	Ctrl時
126*	E1 14 77 E1 F0 14 F0 77	E0 7E E0 F0 7E

＊：このキーはタイプマティックの対象外で，すべての関連するスキャン・コードはキーが押されたときに発生する．

〈表 3-5〉キーボード・コントローラ 8042 の I/O アドレス

アドレス	ビット	R/W	内　容
060h	0～7	R/W	(データ・バッファ)
064h		W	(コマンド・バイト)
	0		割り込み許可 (キーボード)　IRQ1 "1"：許可
	1		予約または，補助装置割り込み許可 (マウス)　IRQ12 "1"：許可
	2		システム・フラグ
	3		(予約済み)
	4		キーボード禁止　CLK 端子を "L" にする　"1"：禁止
	5		IBM-PC モードまたは，補助装置禁止　"1"：禁止
	6		キーボード・スキャン・モード
	7		(予約済み)
064h		R	ステータス・バイト
	0		出力バッファ・フル
	1		入力バッファ・フル
	2		システム・フラグ
	3		コマンド／データ
	4		スイッチ禁止
	5		転送タイムアウトまたは，補助装置バッファ・フル
	6		ジェネラル・タイムアウト
	7		パリティ・エラー

キーボードからパソコンへの送信

キーボードからパソコンへの送信は，コマンドの場合と**データの場合があります**．コマンドといっても，大部分はパソコンからのコマンドに対する応答の意味をもっています (**表 3-3**).

また，データというのはキーの状態をスキャンして得られるスキャン・コードです．スキャン・コードは，キーボードのもつ各キーに対して一意に割り当てられたもので，キーボードのキー構成が変われば定義が変わります．スキャン・コードには 1～3 の 3 種類があり，どれを使うかはパソコンからキーボードへのコマンドで指定できます．スキャン・コード 2 は 84 キーボードに対して上位互換となっています (**表 3-4**).

なお，キーボード側では，SHIFT＋X とか CTRL＋Y というような**複数のキーの組み合わせの認識処理は行いません**．SHIFT キーが押されればそのスキャン・コードをパソコンに転送し，X キーが押されればそのスキャン・コードをパソコンに転送します．

キーボードも，別のデータの転送中を除いて，いつでもパソコンに対してコマンドやデータを送信できます．キーボードの側には転送を中止する権限はないので，DATA および CLK の状態を調べて，転送中でないときだけ CLK を L にして転送を開始します．

キーボード側コントローラ (シングルチップ・マイコン) は，すべてのキーの状態を常時スキャンして，キーの状態の変化 (押し下げ＝MAKE，離し＝BREAK) を検出したら，そのキーに対応するスキャン・コードをパソコンに転送します．ただし，転送を待たされる可能性があるので，一時的にスキャン・コードを保持するための**バッファ**を備えています．

パソコン側では 8042 がキーボードからスキャン・コードを受け取り，システ

データの場合があります
というより，大部分はキー・データの読み出し．たまに，コマンドに対する応答があるという程度．

複数のキーの組み合わせの認識処理は行いません
キーボードは，「SHIFT キーが押された」→「X キーが押された」→「X キーが離された」→「SHIFT キーが離された」などというように，個々のキーの変化を順次パソコンに転送するだけ．これをパソコンの側で，「SHIFT キーが押されている間に X キーが押された」というように判定して，「SHIFT＋X」として認識する．

バッファ
バッファの容量にも限りがあるので，転送が間に合わなくてバッファがあふれたら，オーバラン・コマンドを返す．

ム・スキャン・コードと呼ばれる中間コードに変換します．さらに，システムに対してINT1のハードウェア割り込みを発生して，中間コードをキーボードBIOSに引き渡します．キーボードBIOSは，中間コードをキーに対応するASCIIコードに変換します．なお，Windowsではハードウェア割り込みのINT1を使用せずにキーコードを取得するように変わっています．

システムと8042のインターフェース

8042は，スレーブ・モードのバス・インターフェースをもつちょっと変わったシングルチップ・マイコンとして知られています．アドレスや/RD，/WRなどの制御信号を自分で出力するかわりに，外からの制御信号によってデータ・バスにデータを入出力します．外からは周辺LSIに見えて，中はシングルチップ・マイコンということで，昔はインテリジェントな周辺コントローラLSIを作るのに**たいへん重宝な存在でした**．

したがって，このキーボード・コントローラとして用いられている8042も，システムの中では周辺LSIとして扱われており，I/Oアドレス60h，64hが割り当てられています．60hは8ビット・データ・レジスタ，64hはコマンド/ステータス・レジスタです（**表3-5**）．

このアドレス60hを直接アクセスすれば，システム・スキャン・コードを取得することができます．しかし，実際のアプリケーションではキーボードBIOS（INT 16h）やWindowsのサービスを利用してASCIIコードを取得します．

たいへん重宝な存在でした
最近は，ASICやFPGAを使って周辺コントローラLSIを作るのが一般的．ASICやFPGAにCPUコアを組み込めば，インテリジェントな周辺コントローラLSIも作れる．

◆引用文献◆
(1)吉田功，キーボード&マウス・ポートの構造，トランジスタ技術，1995年10月

トランジスタ技術 SPECIAL　エレクトロニクスの基礎と実用技術を満載したフィールド・ワーク・マガジン

No.71

B5判　160頁　定価1,835円（税込）

特集　OPアンプから始めるアナログ技術
OPアンプ回路の設計からアナログPLDの活用まで

アナログ回路の設計は経験がものをいう世界であるといわれています．ディジタル回路と違って，一朝一夕にアナログ回路技術を身につけるのは難しいのは事実です．しかし，ベテラン技術者もカンと経験だけで回路を設計をしているわけではありません．一つ一つの回路構成，抵抗やコンデンサのパラメータの決定にはそれなりの理由があります．

アナログ回路は先輩技術者が考え出したいくつかの基本回路の動作原理を理解し，最新のデバイスを使った設計のノウハウ…つまり，この回路のこの抵抗値はなぜこの値なのか…を論理的に理解することにより，短期間に設計技術を身につけることが可能です．

ディジタル技術全盛の今日ですが，機器設計の違いは意外にアナログ設計技術の優劣で決まることが多いものです．アナログ技術を身につけて，違いの出せる技術者に飛躍しましょう．

CQ出版社
〒170-8461
東京都豊島区巣鴨1-14-2　販売部　TEL（03）5395-2141　振替 00100-7-10665

第2-4章 キーボード・インターフェースを拡張して作られた
PS/2マウス・インターフェース

宮崎 仁

〈発行日〉
1987年
〈発行者〉
International Business Machines Corporation
〈仕様書〉
PS/2 Mouse Technical Reference

AT互換機のマウス・インターフェース

AT互換機のマウス・インターフェースとしては，シリアル・ポートを利用して接続するシリアル・マウスと，専用ポートをもつPS/2マウスが一般に用いられています．

また，最近ではUSB接続のマウスも普及してきました（**図4-1**）．

もともと，**オリジナルのPC/AT**はマウス・インターフェースを標準装備していませんでした．また，I/Oアドレスや割り込みチャネルもとくにマウス用として用意されたものはありませんでした．当時はマウスを使用するのはCADやゲームなどの特定のアプリケーションに限られており，マウスの必要性はそれほど高くなかったと言えます．

そこで，PC/ATや初期のAT互換機でマウスを使用するには，シリアル・ポートを利用して接続するシリアル・マウスか，ISAに専用拡張ボードを装着して接続するバス・マウスの二つの方法が用いられてきました．バス・マウスはISAスロットを占有してしまいますし，拡張ボードの分だけコストがかかる欠点があり，それほど普及はしませんでした．

シリアル・マウスもシリアル・ポートを占有する欠点がありますが，比較的手軽に使えることから徐々に普及していきました．

しかし，PS/2が設計された頃には**将来のGUI（グラフィック・ユーザ・インターフェース）の発展が視野に入ってきており**，マウスはGUIのポインティング・デバイスの本命と見なされるようになっていました．

そこで，PS/2では従来のキーボード・インターフェースを拡張して，新しいマウス・インターフェースを標準装備するようになりました．これが，PS/2マウスと呼ばれるものです．

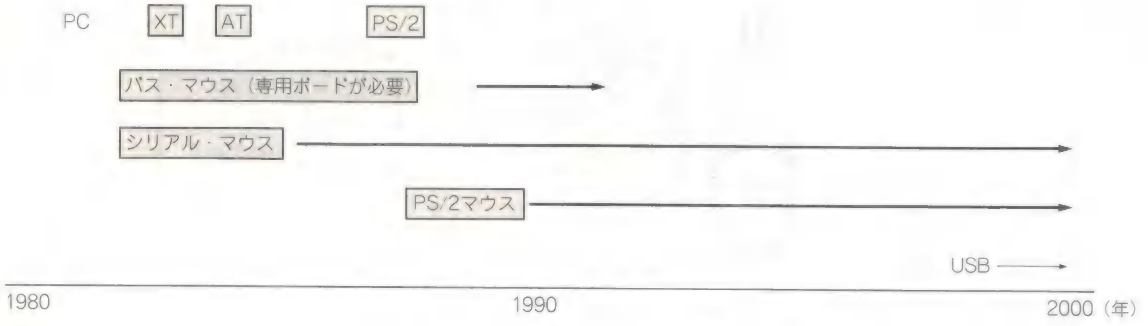
AT互換機ではすでにシリアル・マウスが普及していたことから，PS/2マウス・インターフェースはなかなか普及しませんでした．

しかし，Windowsが標準OSになり，マウスが必須になってくると，シリアル・ポートを占有しないPS/2マウスが人気を集めるようになりました．現在のAT互換機の大部分がPS/2キーボード&マウス・インターフェースを装備しています．

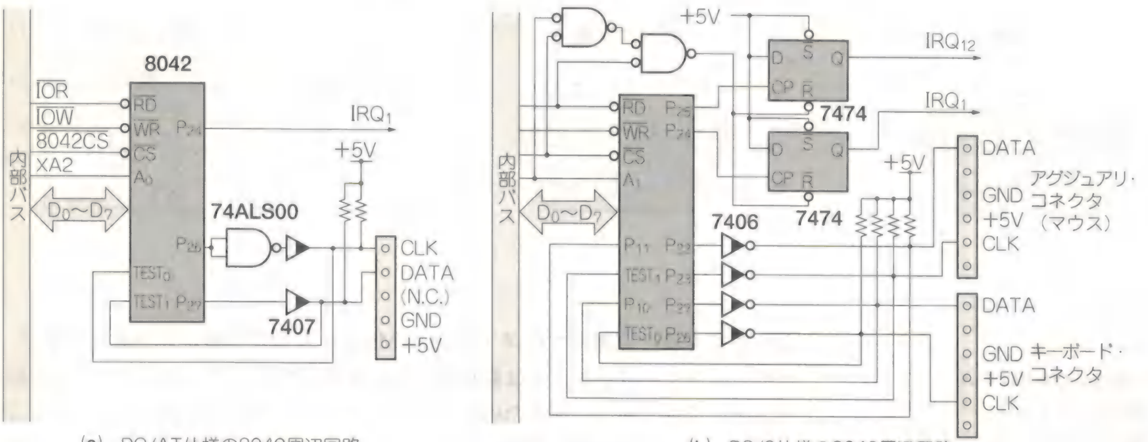
オリジナルのPC/AT
もともとPC/ATでは周辺機能のモジュール化，オプション化を徹底していたので，ディスプレイ・インターフェース，シリアル・インターフェース，パラレル・インターフェース，ディスク・インターフェースなどほとんどの周辺インターフェースは拡張ボード上に搭載されたオプションとして提供されていた．ただ，マウス・インターフェースは純正オプションの中には含まれておらず，Logitech社などのサード・パーティのものが使われていた．

将来のGUI（グラフィック・ユーザ・インターフェース）の発展が視野に入ってきており
1981年発売のIBM PC，1984年発売のPC/ATを設計した時点では，IBM社ではパソコンにGUIを搭載することなど思ってもいなかったようだ．ところが，1984年にApple社から発売されたMacintoshが全世界に衝撃を与えたことから，IBM社でもいやおうなしにGUIの存在を視野に入れなければならなくなった．

〈図 4-1〉AT 互換機のマウス・インターフェース



〈図 4-2〉[1]キーボード・コントローラ 8042 周辺回路



(a) PC/AT仕様の8042周辺回路　　(b) PS/2仕様の8042周辺回路

キーボード・コントローラの空きポートを利用したPS/2マウス

PS/2マウス・インターフェースは，キーボード・コントローラの8042の空きポートを利用して拡張したものです．**ハードウェア上は**，キーボード・インターフェースが2チャネルに増設されたものと見なせます．割り込みチャネルも，従来のキーボード割り込みのIRQ1に加えて，新たにマウス割り込みとしてIRQ12が割り当てられています（**図4-2**）．

コネクタもPS/2キーボードのものと同じ6ピン・ミニDIPです（**図4-3**）．そのため，誤ってキーボードとマウスを逆に挿してしまうこともよくあります．動作はしませんが，壊れることはありません．

なお，PS/2ではマウスのほかにトラック・ボールやタッチ・パッドなどの入力装置もこのインターフェースを利用することを想定しています．これらを総称して，補助装置と呼んでいます．インターフェースは標準装備していても，補助装置そのものはオプションです．

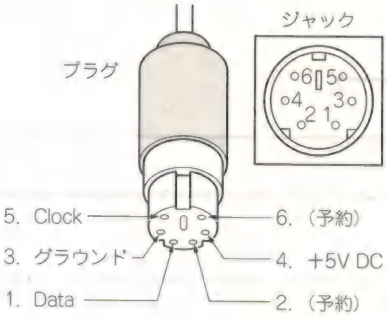
ハードウェア上は

PS/2キーボード・インターフェースとPS/2マウス・インターフェースはハードウェア上は同等であり，ソフトウェアによって一方をキーボード用，一方をマウス用として区別している．

PS/2マウスとシリアル・マウス

PS/2マウスが普及してきたといっても，まだまだシリアル・マウスも広く使われています．これらは互換性はありませんが，マウス本体の機構部や，動きを検出するロータリ・エンコーダ部は共用できますから，コネクタを取り替えるだけ

〈図 4-3〉 PS/2 タイプのキーボード・コネクタ



コネクタのピン割り当て

ピン	I/O	信号名
1	I/O	Data(DATA)
2	−	(Reserved)
3	−	グラウンド
4	−	+5V DC
5	I/O	Clock(CLK)
6	−	(Reserved)

〈図 4-4〉 シリアル・マウスのコネクタ

コネクタのピン割り当て例

ピン	I/O	信号名
2	I	RxD (DATA)
3	O	TxD (−10V)
5		SG (GND)
7	O	RTS (+10V)

見た目ではわかりにくい
単売のマウスなら，パッケージや説明書に両用と書いてあるか，変換コネクタが付属しているものは両用マウスと判断できる．だが，マウス自体には何も書いていないものが多い．

相性の問題
とくにシリアル・マウスやバス・マウスは方式が統一されているわけではないので，独自のドライバを組み込まないと動作しないものがあったり，あるいは似たようなドライバがすでに組み込まれていればそれで動いてしまう場合もある．逆にそれらが干渉して不具合を起こすこともある．また，ソフトウェアでなくてハードウェアの相性の問題もある．

パソコンからマウスへの電源供給
マウスの消費電力は一般にごくわずかだが，それでも0では動かない．PS/2 マウスなら専用の電源線（5 V, GND）をもっているが，シリアル・ポートにはもともと電源供給のための線はない．信号線からマウスに電源を供給するように工夫されている．

でどちらにも使える両用マウス（Combo Mouse）も数多く製品化されています．

ただ，両用マウスかどうかは**見た目ではわかりにくい**ので，あるマウスは変換アダプタを付けて両方に使えたのに，あるマウスは一方にしか使えないということで，ユーザの混乱を招く原因にもなっているようです．そうでなくても，マウスはパソコン本体やOS，各種のドライバやアプリケーションとの**相性の問題**が多い装置であり，トラブルの話はよくあります．

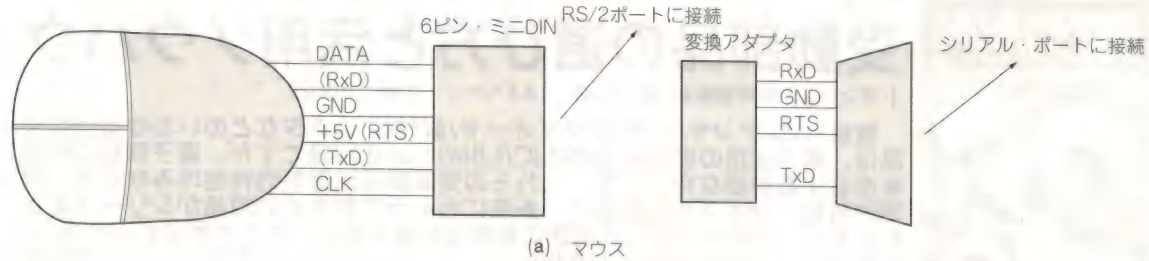
シリアル・マウスは，シリアル・ポートを通じてデータをパソコンに送ります．9 ピンのシリアル・ポート（p.128）のうち，一般に TxD（送信データ），RxD（受信データ），RTS（送信要求），SG（信号グラウンド）の 4 本の線を使います（**図 4-4**）．なお，この送信，受信というのはパソコン（DTE）を中心に考えたときの機能です．マウスから見れば，RxD が送信になります．

マウスが動いたりボタンをクリックしたときに，RxD を使ってパソコンにそのデータを送ります．それに対して，パソコンからマウスに対してデータを送る必要はありません．TxD はデータ線としてではなく，**パソコンからマウスへの電源供給**に使われています．シリアル・ポート（RS-232-C）の信号電圧は約 ±10 V ですが，パソコン側は TxD 出力を使って −10 V，RTS 出力を使って +10 V をマウスに供給します．また，RTS 出力はパソコンからマウスに合図を送るためにも使います．

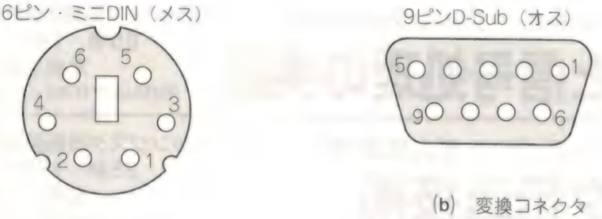
パソコンによっては，このような電源供給能力をもたないため，シリアル・マウスを接続しても正常に動作しないものもあります．また，最近の省電力機能をもつパソコンでは，動作中にマウスに対する電源供給が止まってしまうものもあります．

また，PS/2 マウスはパソコンから 5 V 電源を供給されて動作しますから，両用でない PS/2 マウスを変換アダプタでシリアル・ポートにつないでしまうと，最悪の場合マウスを壊してしまうこともあります．両用マウスは，どちらの電

〈図 4-5〉両用マウスの例



(a)　マウス



変換アダプタのピン割り当て

6ピン	9ピン	信号名
2	2	RxD(DATA)
3	5	SG(GND)
4	7	RTS(＋10V)
6	3	TxD(−10V)

(b)　変換コネクタ

源電圧でもうまく動作するように，何らかの工夫がされています．

もう一つの問題は，PS/2マウスはPS/2キーボードと同じクロック同期式のシリアル・インターフェースなのに対して，シリアル・マウスは調歩同期式のシリアル・インターフェースです．電圧が違うだけでなく，データの送り方も違います．

そこで，たとえばマウス自体に2種類のインターフェース機能とPS/2コネクタを付けておき，PS/2コネクタのうち使っていない2本のピン(2ピンと6ピン)をシリアル・インターフェースとの接続に利用する方法が使われています(**図4-5**)．

また，それとは異なる方法として，マウス自体にはクロック同期式のインターフェース機能だけをもたせて，シリアル・ポートのハンドシェイク線（RTS/CTSやDTR/DSR）を利用してデータ入出力を行うという例もあるようです．

いずれにしても，シリアル・マウスとPS/2マウスの変換にはかなりイレギュラーな方法が用いられており，いつでもうまくいくというわけではありません．

◆引用文献◆

(1)吉田功；キーボード&マウス・ポートの構造；トランジスタ技術；1995年10月

トランジスタ技術 SPECIAL No.51
エレクトロニクスの基礎と実用技術を満載したフィールド・ワーク・マガジン
B5判　160頁
定価 1,835円（税込）
特集　データ通信技術基礎講座
RS232Cの徹底理解からローカル通信の実用技術まで

CQ出版社
〒170-8461
東京都豊島区巣鴨1-14-2　販売部　TEL(03)5395-2141　振替 00100-7-10665

ハードウェア・デザイン・シリーズ　　好評発売中

抵抗，コンデンサ，コイル，ダイオード，線材＆コネクタの機能と特徴

受動部品の選び方と活用ノウハウ

トランジスタ技術編集部 編・B5判・144ページ・定価1,800円

　抵抗/コンデンサ/コイル/ダイオード/線材/コネクタなどのいわゆる受動部品は，電子回路の機能を決めるIC/LSIのための脇役ですが，電子回路の性能を左右する重要な存在です．これらの受動部品は電気的特性のみならず，物理的特性，サイズや重量など，多肢にわたって数多くの製品がシリーズ化されており，回路設計者には回路に最適な品種を選択することが求められます．本書では，各種受動部品の選択のポイントを，その特徴を生かした応用回路の具体例を示して解説しています．

A-D/D-Aコンバータの接続方法からディジタル・フィルタの実現まで
DSPのハードウェアと信号処理の実際
トランジスタ技術編集部 編
B5判　144ページ
定価1,800円
ISBN4-7898-3275-9

微弱電波/特定小電力によるデータ伝送からスペクトラム拡散まで
電波による無線データ伝送技術
トランジスタ技術編集部 編
B5判　128ページ
定価1,400円
ISBN4-7898-3426-3

発光ダイオードからフォト・カプラ，赤外線，光ファイバの応用まで
光エレクトロニクスの基礎と活用法
トランジスタ技術編集部 編
B5判　162ページ
定価1,680円
ISBN4-7898-3425-5

シリアル・ポートとパラレル・ポートを活用しよう
パソコン・アダプタの製作＆応用
トランジスタ技術編集部 編
B5判　144ページ
FD付き　定価1,800円
ISBN4-7898-3424-7

アート・ワークの基礎から多層基板の設計事例まで
パソコンによるプリント基板設計
中島 直樹　著
B5判　176ページ
CD-ROM付き　定価 2,200円
ISBN4-7898-3423-9

部品がわかればハードウェア技術がわかる
わかる電子回路部品 完全図鑑
トランジスタ技術編集部 編
B5判　160ページ
フルカラー　定価1,890円
ISBN4-7898-3422-0

3端子/チョッパ/フライバック各種レギュレータICの使い方
電源用IC活用マニュアル
トランジスタ技術編集部 編
B5判　160ページ
定価1,682円
ISBN4-7898-3420-4

抵抗，コンデンサ，インダクタ，機構部品の特徴と仕様
わかる電子部品の基礎と活用法
蓟 利明/竹田 俊夫 共著
B5判　184ページ
定価1,733円
ISBN4-7898-3416-6

受動部品/機構部品を100%活用するために
電子回路部品活用ハンドブック
トランジスタ技術編集部 編
B5判　288ページ
定価1,886円
ISBN4-7898-3170-1

整流回路からスイッチング・レギュレータまで
実用電源回路設計ハンドブック
戸川治朗 著
B5判　240ページ
定価1,998円
ISBN4-7898-3168-X

CQ出版社　販売部　郵便振替 00100-7-10665　e-mail：sales@cqpub.co.jp

※定価は消費税込みの価格です

CQ出版社のURL　http://www.cqpub.co.jp/

定本シリーズ　　好評発売中

高速・高信頼ディジタル・システムのための設計ノウハウ

定本 ASICの論理回路設計

小林芳直 著

A5判　344ページ　定価2,730円

ISBN4-7898-3044-6

ASIC…Application Specific Integrated Circuit＝特定用途向けICは，一般に市販されている汎用ICではなく，ユーザが特定の用途に使用するために作るLSIです．

本書は1988年に出版された「ASICの論理回路設計法」を基本にして全面改訂したものです．当時はCADも十分に発達しておらず，PLDが普及しはじめ，FPGAはやっとXILINX社のものが登場した頃でした．完全同期式回路の理論的な良さは理解されていても，現場ではまだまだ少ない存在でした．現在ではPLDやFPGAの設計は同期式が当然になってきましたし，ASICでもVHDLで書けば同期式回路を作ることができるようになってきました．

そんな大きく変化した時代に対応したのが本書です．

ASICを作るためには，まず特定用途の機能を実現する論理回路を用意しなくてはいけません．本書では，ASICの大規模な論理回路を同期式回路で実現するための基礎知識と実用ノウハウ，ASICを実現する上での高信頼性化技術についても詳しく解説しています．座右においてASIC設計のバイブルとしてお役立てください．

ASIC＝同期式設計＝高信頼性設計への誘い／ゲート回路のいろいろ／論理圧縮とクワイン・マクラスキー法／コンビネーション回路の設計／カウンタ回路の設計／ジョンソン・カウンタの設計／シーケンサの設計／シーケンサの高信頼性化技術／シーケンサの応用設計／最適設計へのアプローチ

論理回路の記述からステート・マシンの設計まで

定本 ASICのシステム設計

小林 芳直 著
A5判　272ページ
定価 2,548円
ISBN4-7898-3045-4

---

*CR*発振からディジタル・シンセまでを実験で解析

定本 発振回路の設計と応用

稲葉 保 著
A5判　336ページ
定価 2,854円
ISBN4-7898-3046-2

増幅回路技術を実験を通してやさしく解析

定本 トランジスタ回路の設計

鈴木 雅臣 著
A5判　324ページ
定価 2,243円
ISBN4-7898-3048-9

FET/パワーMOS/スイッチング回路を実験で解析

定本 続トランジスタ回路の設計

鈴木 雅臣 著
A5判　360ページ
定価 2,752円
ISBN4-7898-3047-0

ディジタル技術の基礎からASIC設計まで

定本 ディジタル・システムの設計

猪飼 國夫/本多 中二 共著
A5判　452ページ
定価 3,059円
ISBN4-7898-3049-7

再現性を重視した設計の基礎から応用まで

定本 OPアンプ回路の設計

岡村 廸夫 著
A5判　424ページ
定価 2,854円
ISBN4-7898-3050-0

CQ出版社　販売部　〒170-8461 東京都豊島区巣鴨1-14-2　☎(03)5395-2141　FAX(03)5395-2106

※定価は消費税込みの価格です

CQ出版のURL　http://www.cqpub.co.jp/

編集雑記

編集部から

● 最近の新聞の社会面を騒がせているのは，雪印乳業や三菱自動車工業といった，大企業の不祥事です．とくに，テレビ放送の入った記者会見での経営陣の醜態は，報道のとおりですが，当の企業に勤めている人たちの思いやいかばかりでしょう．

● 売り上げ第一，コスト優先，官僚主義がまねく弊害です．だれに対して商売をしているのかわからなくなっているのでしょう．自戒を込めて，欠陥商品を作らないようにしていきたいと思います．と，M自動車のクルマを購入して，みごと欠陥車に当たり，欠陥3点セットのリコールを受け，牛乳，チーズ，バターと，すべて雪印製品の愛用者の私にとっては，もう大丈夫だと思っているのですが….

● この号が発売されるころには，シドニー・オリンピックも終わりに近づき，日本のメダルの数も確定しているのではないかと思います．スポーツにおけるアマチュアリズムはとうの昔になくなってしまいました．東京オリンピックをオンタイムで見た者にとってはなつかしく思います．

● 企業がスポーツを支えるのではない，新しい方策が必要です．雪印乳業のジャンプの選手や三菱自動車工業の野球の選手が責任をとることはないのです．メダルを取ればいいというものではないでしょうが，その国のメダルの数がそのイベントそのものを盛り上げる力があるのも事実です．

● 今回の特集は，パソコンにまつわる規格のお話ですが，創業者利益を得ようと，早く市場に投入して，広まったところで特許料をガッポリいただこうという戦略が見えます．市場つまりお客さんによろこんで使ってもらわなくてはもとの黙阿弥になってしまいます．（檀）

● 音質はともかく，MP3対応のCDプレーヤーの出現でCDチェンジャーの存在が…．約1/10に圧縮されると言うことは，約10枚の音楽CDをMP3で圧縮すれば1枚のCD-Rにやけるということですよネ！あッ！MP3対応のCDチェンジャーが出るかも？（MASA）

● トランジスタ技術SPECIALの既刊号で紹介しました基板等の頒布サービスを，申し込み締め切り日を過ぎて受け付けているものがありますのでお知らせします．No.20のMICRO-CAP III，No.23のPALライタ基板，PALASMソフト，No.29のZ80マイコン・キット，No.38の拡張I/Oモジュール・キット，No.55の基板，部品キット，No.57の電源基板キット（第5章を除く）です．申し込み方法は各雑誌掲載のとおりです．

● 本誌掲載記事の利用についてのご注意――本誌掲載記事には著作権があり，また工業所有権が確立されている場合があります．したがって，個人で利用される場合以外は所有者の承諾が必要です．

また，掲載された回路，技術，プログラムを利用して生じたトラブルなどについては，小社ならびに著作権者は責任を負いかねますのでご了承ください．

● ご質問はお手紙で――本誌掲載記事に関する技術的なご質問は，往復はがきか，返信用封筒を同封した書簡を編集部あてお寄せください．執筆者に回送し，直接回答していただきます．質問の内容は当該記事を逸脱しない範囲で，できる限り具体的に明記してください．また，お電話によるご質問にはお応えできませんので，ご了承ください．

次号のお知らせ(12月27日発売)

特集　家電，情報機器を支える ブラシレス・モータの制御技術

フロッピ・ディスクはもちろん，ハードディスクやCD & CD-R，DVD，さらにレーザ・スキャナなど，情報機器を支える記憶・再生装置には大量のブラシレス・モータが使われています．このブラシレス・モータの制御回路技術について詳しく解説をします．ご期待ください．

トランジスタ技術 SPECIAL エレクトロニクスの基礎と実用技術を満載したフィールド・ワーク・マガジン No. 72
発行所　CQ出版株式会社（無断転載を禁じます）
〒170-8461　東京都豊島区巣鴨 1-14-2
電　話　編集部：03(5395)2121　広告部：03(5395)2133
販売部：03(5395)2141
振　替　00100-7-10665
編集人　山形孝雄
発行人　蒲生良治
Printed in Japan
© CQ出版株式会社 2000（定価は表四に表示してあります）
2000年10月1日発行　制作：クニメディア㈱　印刷・製本：三晃印刷㈱

郵便はがき

料金受取人払

豊島局承認

7198

差出有効期間
2001年11月30
日まで

切手をはらずに
お出しください

170-8790

（受取人）

東京都豊島区巣鴨1-14-2

CQ出版株式会社

トランジスタ技術SPECIAL リーダーズカード 行

No.72

■編集部への通信欄（自由なご意見をお書きください）

資料請求カード　SP.No.72

当社記入欄　1 VOL 0072　カード有効期限：2000年12月末日迄

2 受付

3 登録内容変更箇所No.

4 請求状況　1＝新規　2＝登録済　3＝登録内容の変更

5 会員番号 CQ
※会員の方のみご記入してください。

6 フリガナ

7 姓名　姓　名

8 生年月日　19　年　月　日

E-mail：

9 〒　—　都道府県　区市郡　区町村

10 勤務先住所

11 会社名

12 部署名

13 勤務先TEL　市外局番　局番　電話番号　(　)

20 勤務先FAX　市外局番　局番　電話番号　(　)

14 役職　15 職種　16 専門分野

17 業種　18 従業員数

19 広告資料請求欄　(チェック欄に✓を記入ください)

チェック欄	請求No.	広告主名
	0002	石塚電子㈱
	0003	㈱ケーアイテクノロジー
	0004	バイコージャパン㈱
	0005	㈱ESP企画

資料請求カード記入法

専門誌の広告は重要な技術情報です。その中で読者の皆様が入手したい情報は、資料請求カードのご利用で、居ながらにしてお手元に届きます。ぜひ有効にご活用下さい。

- 必ず全項目にご記入下さい。
- 姓名・勤務先・電話番号・住所は所定の様式通り1枠ごとに書き入れて下さい。
- 会社名は株式会社ならば(株)、有限会社は(有)、財団法人は(財)と略し社名を書いて下さい。
- 生年月日は西暦で書いて下さい。
- 当カードは登録することができます。まず全項目にご記入の上投函下さい。当社より登録完了はがきが到着します。
- 当社請求カードの会員番号がある方は、会員番号・姓名・生年月日の記入ですみます。また会員番号をお持ちで登録内容に変更がある方は変更箇所を上記の他に書いて下さい。
- 役職・所属部門・担当分野・業種・従業員数は下記の該当する番号を記入して下さい。

役職
1. 一般職
2. 主任職
3. 係長職
4. 課長職
5. 部長／次長職
6. 役員職
7. 経営者
8. 教職(小、中、高校)
9. 教職(大学、専門学校)
10. 専門職(弁護士・医師・会計士等)
99. その他

職種
1. 研究・開発部門
2. 設計・技術部門
3. 生産・生産管理部門
4. 保守サービス部門
5. その他の技術部門
6. 営業部門
7. 購買・資材部門
8. 総務・経理部門
9. 人事・教育部門
10. 経営部門
99. その他

専門分野
1. コンピュータ(ハードウェア)技術
2. コンピュータ(ソフトウェア)技術
3. 半導体・デバイス技術
4. 電子回路技術
5. 計測・制御技術
6. 通信関連技術
7. 物性・材料
8. 機構設計
9. システムエンジニアリング
10. 検査・品質管理
11. 医師
12. 経営管理
13. 商工サービス
14. 教職
15. 学生
99. その他

業種
1. 建設・不動産
2. 金融・保険
3. 化学・エネルギー
4. 鉄鋼・金属
5. 機械・精密機器
6. 電子機器・部品
7. 輸送用機器・運輸
8. その他の製造
9. 商社・卸売
10. 通信・情報
11. 情報処理
12. その他のサービス
13. 政府公共機関
14. 学校・研究所・病院
15. 農林・水産業
99. その他

従業員数
1. 10人以下
2. 11人～50人
3. 51人～100人
4. 101人～500人
5. 501人～1000人
6. 1001人～5000人
7. 5001人～10,000人
8. 10,001人以上

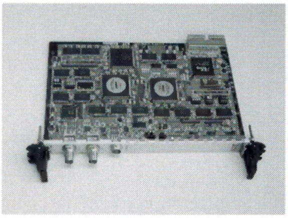
画像処理とPCI

KITの専門は画像処理であり、各開発目的は、画像処理技術を応用した各種ハードウェアの開発です。しかしその実現手段として、急激に広まっているパーソナルコンピュータやPCIボードを利用する事も、コスト面からとても重要になっております。現実に近年の開発テーマを振り返りますと半数近くがPCI（又はCompactPCI）ボードです。その他としてはスタンドアロンの専用機やVME、ISA専用バス等もあります。そこで手段として最も多く利用してきたPLXを用いたPCIボード設計部をオープンリリースとして広く販売致します。もちろん本業は「アイデア豊富なカスタム設計」ですので弊社内でもこれを利用したカスタム品を開発致します。「画像処理で困ればKIT」としておもしろい開発テーマを待っております。

技術力評価は、自社ブランドで!!

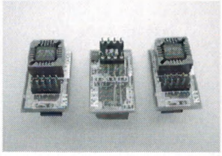
KIT1010 (CompactPCI画像処理ボード)

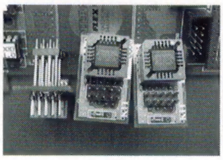
画像処理チップIP00C101の多芸を知って頂く為にキャンペーン価格を実施します。　¥198,000（税別）



KIT1020 (EPC EMU) 30%OFFキャンペーン中

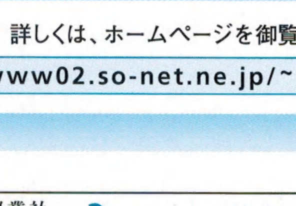
多くのユーザに利用して頂きましたので、1個からキャンペーンプライスで販売します。1度御利用頂きますと手放せなくなります。

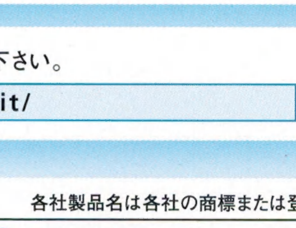




KIT1030 (PLX Getter)

「手段」としてPCIを用いる人は、「安定動作」が重要だと思います。PCIバスタイミングは、とてもシビアなので、そこは専用のPLXに任せ、ユーザは荷加機能に専社しましょう。サンプル回路図やドライバソースも付けたセミナーを実施します。





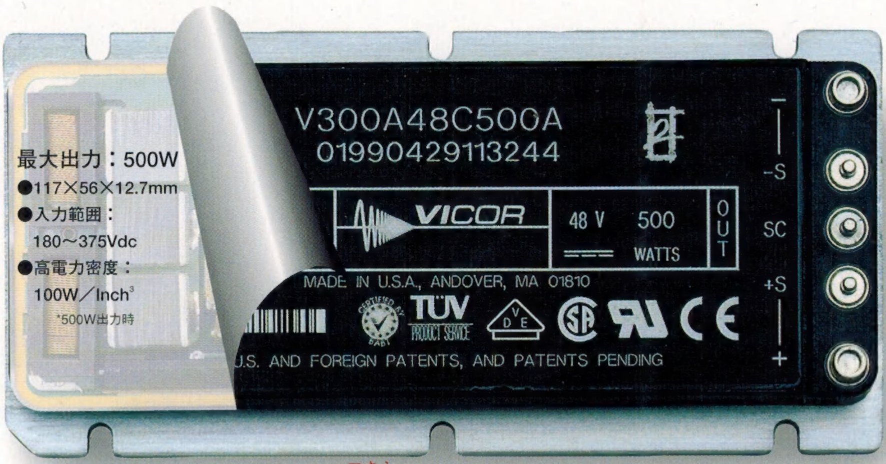
セミナー開催！ 詳しくは、ホームページを御覧下さい。

http://www02.so-net.ne.jp/~kit/

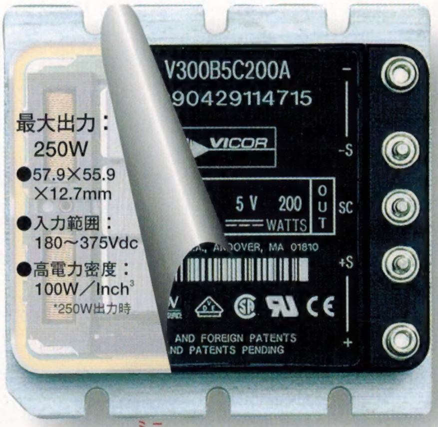
各社製品名は各社の商標または登録商標です。

Knowledge Imaging Technology KIT システム開発専門業社 株式会社ケーアイテクノロジー
〒224-0032 神奈川県横浜市都筑区茅ヶ崎中央19-1 タウンスクエア3階 TEL：045（949）5331（代） FAX：045（949）5332

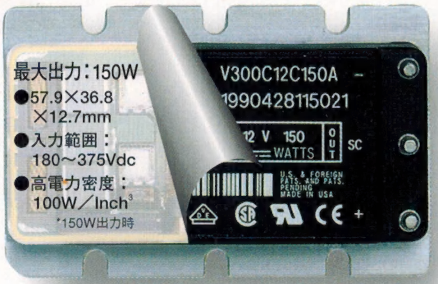
その正体は次代の、コンバータモジュールである。



マキシ
Maxi Family



ミニ
Mini Family



マイクロ
Micro Family

原寸大

100 W/inch³ という高電力密度。
91% の高効率。
1/3 の部品点数。（弊社比）

2nd Generationシリーズ
300V Input Family

Component Solutions For Your Power System

VICOR Japan Co.Ltd
バイコー ジャパン株式会社
〒141-0031 東京都品川区西五反田8-9-5 ポーラ第3五反田ビル6F
TEL:03-5487-3884 FAX:03-5487-3885
http://www.vicr.co.jp

CQ出版社
〒170-8461 東京都豊島区巣鴨1-14-2
☎(03)5395-2141(販売部)

定価1,840円
本体1,752円

雑誌 16711-10

資料請求 No.4

T1116711101840